



www.phpbuch.cc

Umfassend zur
**MySQL-
Programmierung**

Jörg Krause

Grundlagen und Profiwissen

PHP 4

Webserver-Programmierung unter Windows und Linux

2., überarbeitete
Auflage

GERBER

HANSER



V

S

I

1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E

Vorwort

PHP verspricht Skriptprogrammierung für jedermann. Die große Kunst bei einem umfassenden Fachbuch zu diesem Thema besteht darin, es nun jedermann Recht zu machen. Dieser Auflage gingen bereits zwei weitere voraus, ein Buch zu PHP 3 und eines zu PHP 4 – kurze Zeit nach Fertigstellung der ersten Version 4.0.0 erschienen. Das Konzept scheint dabei die Interessen der größtmöglichen Gruppe Jedermannen und Jederfrauen getroffen zu haben. Dies beweisen zahllose begeisterte Leserbriefe und die Tatsache, dass »PHP 4 – Grundlagen und Profiwissen« der mit Abstand erfolgreichste deutschsprachige PHP-Titel ist.

Das Konzept wurde beibehalten, der Inhalt stark überarbeitet. So hat sich PHP – trotz vergleichsweise kleiner Versionssprünge – kräftig weiterentwickelt. Dadurch kam es bei einigen Skripten zu Änderungen, die berücksichtigt werden mussten. Insgesamt wurde die Struktur etwas entzerrt und vereinfacht. Die »Grundlagen« und theoretischen Ausführungen sind in Kapitel 2 zusammengefasst und der praktische Teil konnte umfassenreicher und inhaltsreicher gestaltet werden. Drastisch zugenommen hat die Anzahl der Beispiele. Die Referenz wurde dagegen weitgehend verkürzt und auf eine Kurzreferenz reduziert. Das ist kein Mangel, denn vor Erscheinen dieser Auflage ist bereits eine gedruckte PHP-Referenz bei Carl Hanser erschienen – weitaus umfangreicher und mit mehr Beispielen versehen, als hier Platz gewesen wäre. Zudem hat auch die deutschsprachige Ausgabe des offiziellen Manuals deutlich zugelegt, sodass ein erneuter Abdruck wenig sinnvoll erschien. Der Umfang hat sich dennoch nicht verändert, was darauf hinweist, dass der praktischen Lehre von PHP größtmöglicher Raum eingeräumt werden konnte. Auch in dieser Hinsicht ist das Buch einzigartig: Mehr PHP gibt es nirgends.

Neu in diesem Buch ist ein umfassendes Projekt zur Vereinfachung der Programmierung von Webseiten: *phpTemple*. Damit können auch unbedarfte Webdesigner schnell und sicher komplexe Seiten entwerfen, basierend auf PHP. Die Vorstellung in diesem Rahmen soll auch eine Aufforderung sein, sich mit der Thematik Trennung von Design und Funktion intensiver zu beschäftigen. Der Leser wird so auf sehr sanfte Weise an die Techniken professioneller Siteprogrammierung herangeführt. Damit wird der Aspekt »Profiwissen« stärker betont.

Da auch dieses Buch kein Endprodukt ist und auf Auflage 2 sicher Auflage 3 folgt, sei der Leser auch hier wieder dazu aufgefordert, jeglichen Kommentar an die bekannte Adresse zu senden: joerg@krause.net. Scheuen Sie sich nicht, auch mal ein erfolgreiches Projekt vorzustellen, vielleicht finden Sie es in einer künftigen Auflage als Anregung wieder.

Jörg Krause

Berlin, im September 2001

Vorwort zur ersten Auflage

Mit der Version 4 ist PHP endgültig zu einer reifen und umfassenden Programmiersprache für das Web geworden. Der Umfang der Sprache lässt den Schluss zu, dass PHP die führende Sprache im Web werden kann. Eine klare, einfach zu verstehende Syntax und starke Funktionen mit hoher Spezialisierung sind der beste Weg, auch ungeübte »Quereinsteiger« schnell zu Erfolgen zu führen.

Eine komplexe Sprache verlangt nach einer klaren Darstellung. Das Online-Handbuch ist, zumal nur zögerlich übersetzt, den meisten keine besonders gute Hilfe. Mit dem vorliegenden Buch habe ich versucht, die Sprache vollständig und bedarfsgerecht aufzubereiten. Zum einen ist der einführende Teil weiter ausgebaut worden, zum anderen ist die Referenz nun tatsächlich vollständig (was auch exotische Funktionen betrifft). Die Systematik wurde weiter verbessert, eine Vielzahl von Querverweisen erleichtert die Navigation, vor allem auch in der Referenz. Kapitel- und Abschnittsanfänge sind noch konsequenter mit einer einleitenden Überschrift versehen worden. So kann man sich oft das »Anlesen« auf der Suche nach bestimmten Funktionen ersparen. Im Profiteil kamen einige Beispiele hinzu, die den Einstieg vor allem für »fast« Fortgeschrittene erleichtern und den Anspruch eines Lösungsbuches unterstreichen – letztendlich ist die fertige Applikation in kürzester Zeit das Ziel – Abschreiben ist ausdrücklich erwünscht.

PHP 4 ist nun vollständig und vor allem auf Grund der Final Version implementiert. Im Gegensatz zu Beta-Versionen wird hier die Darstellung der Endversion gezeigt, wie sie auch bei Providern im Einsatz ist. Trotzdem wird die Umstellung nur zögerlich vonstatten gehen, vor allem auch, weil einige Skripte geändert werden müssen und Provider kaum zwangsweise eine Umstellung verfügen. Die Realität ist also eine langjährige Koexistenz zwischen PHP 3 und PHP 4. In diesem Buch wird diese Koexistenz ebenso abgebildet. Ohne Kennzeichnung sind die gezeigten Funktionen in beiden Versionen verfügbar. Was in PHP 4 neu hinzukam, wird durch das PHP4-Symbol gekennzeichnet. Was in PHP 4 nicht mehr gültig ist, bekam das PHP 3-Siegel. Diese Kennzeichnung finden Sie natürlich auch in der Referenz. Sie können das Buch deshalb auch uneingeschränkt als PHP 3-Buch ansehen.

Eine sehr emotionale Diskussion wird immer wieder um die Plattformfrage geführt. Hier gibt es sicher verschiedene Szenarien. Was immer Sie tun, nehmen Sie die Plattform, auf der Sie sich »zu Hause« fühlen. Ich habe die am häufigsten verwendete Plattform-Kombination als Grundlage zum Schreiben genommen: Windows als Entwicklungsumgebung, Linux für den Webserver. Falls Sie Linux zur Entwicklung nutzen, werden Sie selbst besser wissen, wie Sie damit umgehen.

Ich hoffe, Sie sind damit gut auf PHP 4 eingestimmt und können schnell gute Skripte entwickeln. Für Feedback jeder Art gilt unverändert der E-Mail-Kontakt: joerg@krause.net.

Jörg Krause

Berlin, im November 2000



V

S

I

1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E

Schnellübersicht Kapitel

Vorwort zur zweiten Auflage	5
Schnellübersicht Kapitel.....	7
Inhaltsverzeichnis	9
Teil I – Einführung und Grundlagen.....	21
1 Einführung.....	23
2 Vorbereitung und Installation	45
3 Erste Schritte mit PHP	145
4 Interaktive Webseiten	299
5 Praxis I – Lösungen für den Alltag	423
Teil II - Datenbankprogrammierung	453
6 Grundlagen der Datenbanktechnik	455
7 Datenbankprogrammierung	529
8 Praxis II – Datenbanklösungen	597
Teil III – PHP professionell programmieren.....	627
9 Fortgeschrittene Programmierung	629
10 Zusatzprogramme	747
11 Praxis III – Das PHP-Projekt	767
Teil IV – Anhänge und Referenz	847
A Glossar	849
B Kurzreferenz	883
C Server-Variablen und Statuscodes	1093
D Index	1105
E An den Autor.....	1125



Inhaltsverzeichnis

Vorwort zur zweiten Auflage	5
Schnellübersicht Kapitel	7
Inhaltsverzeichnis	9
 Teil I – Einführung und Grundlagen	21
1 Einführung	23
1.1 Was ist PHP?	25
1.2 Zielgruppe	27
1.3 Wie dieses Buch zu lesen ist	28
1.3.1 Aufteilung	28
1.3.2 Kapitelübersicht	29
1.3.3 Icons und Symbole	29
1.3.4 Schreibweisen und Satz	30
1.4 Die Buch-CD und Website	32
1.4.1 Allgemeines	32
1.4.2 Quellen im Internet	32
1.5 Dynamische Webseiten	36
1.6 Linux oder Windows?	37
1.7 Was ist Open Source?	42
1.8 Tipps zum sauberen Programmieren	43
2 Vorbereitung und Installation	45
2.1 Netzwerkgrundlagen	47
2.1.1 Rückblick	47
2.1.2 Ausblick	48
2.1.3 Grundbegriffe	49
2.1.4 Die Internet-Protokolle und ihr Ursprung	53

V
S

I

1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E

2.1.5	Die Organisationen ICANN und IANA.....	53
2.1.6	Das ISO/OSI-Modell und die Internet-Protokolle.....	54
2.1.7	Request For Comments (RFC)	57
2.1.8	Internetprotokolle im Detail.....	59
2.2	Höhere Netzwerkprotokolle	72
2.2.1	HTTP	72
2.2.2	File Transfer Protocol (FTP)	77
2.2.3	Protokolle für E-Mail-Verkehr	84
2.2.4	Grundlagen der E-Mail-Programmierung	89
2.2.5	Network News Transfer Protocol (NNTP)	104
2.3	Vorbereitung der Installation.....	106
2.3.1	Beschaffung der Komponenten	106
2.4	Installation unter Windows NT/2000.....	107
2.4.1	WAMP	107
2.4.2	WIMP	110
2.4.3	WXMP	113
2.5	Installation unter Linux.....	114
2.5.1	Vorhandenes System nutzen	114
2.6	PHP konfigurieren	119
2.6.1	Die Datei php.ini.....	119
2.7	Sicherheit	135
2.7.1	Sicherheitsprobleme	135
2.7.2	Angriffsszenarien	136
2.8	PHP im Webpace.....	137
2.8.1	Vorgegebene Daten	138
2.8.2	Überprüfen der Konfiguration	139
3	Erste Schritte mit PHP	145
3.1	Aufbau der Skripte	147
3.1.1	Wo wird programmiert?.....	147
3.1.2	PHP und HTML.....	147
3.1.3	Dateien einschließen	150
3.1.4	PHP und JavaScript.....	151
3.2	Eine kompakte Einführung in PHP.....	151
3.2.1	Ausgabe von Daten an den Browser	152

3.2.2	Kommentare.....	154
3.2.3	Variablen.....	157
3.2.4	Datentypen.....	161
3.2.5	Konstanten.....	168
3.2.6	Operatoren.....	170
3.2.7	Arrays.....	173
3.2.8	Zeichenkettenverarbeitung.....	189
3.2.9	Formatierungsfunktionen.....	201
3.2.10	Mathematische Funktionen.....	204
3.2.11	Zeit- und Datumsfunktionen.....	207
3.2.12	Reguläre Ausdrücke.....	217
3.2.13	Webspezifische Funktionen.....	228
3.3	Programmieren mit PHP.....	230
3.3.1	Blöcke und Strukturen.....	230
3.3.2	Bedingungen.....	231
3.3.3	Schleifen.....	238
3.3.4	Benutzerdefinierte Funktionen.....	245
3.4	Objektorientierte Programmierung.....	254
3.4.1	Einführung.....	254
3.4.2	Definition einer Klasse.....	256
3.4.3	Erweiterte Techniken für Objekte.....	258
3.4.4	Spezielle Funktionen.....	259
3.4.5	Praktischer Umgang mit Klassen.....	264
3.4.6	COM-Objekte.....	268
3.5	Fehlerbehandlung.....	271
3.5.1	Abbruchsteuerung.....	271
3.5.2	Das Fehlerkonzept in PHP.....	272
3.5.3	Umgang mit Laufzeitfehlern.....	273
3.5.4	Benutzerdefiniertes Fehlermanagement.....	274
3.5.5	Fehler vermeiden.....	278
3.5.6	Hilfe bei der Fehlersuche.....	285
3.5.7	Debugging-Tipps.....	286
3.6	Hilfsfunktionen.....	294
4	Interaktive Webseiten.....	299
4.1	Formulare auswerten.....	301

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

4.1.1	GET und POST	301
4.1.2	Daten aus einem Formular ermitteln.....	303
4.1.3	Formularelemente auf Existenz testen	307
4.1.4	Formulare und JavaScript	308
4.1.5	Komplexe Formulare	314
4.2	Daten per URL weiterreichen.....	318
4.2.1	Wie werden Daten weitergereicht?.....	318
4.2.2	Probleme	320
4.3	Cookies	324
4.3.1	Einführung in Cookies.....	324
4.3.2	Cookies in PHP	329
4.3.3	Cookies und JavaScript.....	333
4.4	Verbindungssteuerung.....	333
4.4.1	Funktionsübersicht.....	333
4.4.2	Einführung	334
4.4.3	Ausführungssteuerung.....	336
4.5	Sessionverwaltung	336
4.5.1	Einführung in die Sessionverwaltung	336
4.5.2	Sessions mit versteckten Feldern.....	338
4.5.3	Sessions mit URI	340
4.5.4	Sessions mit Cookies	340
4.5.5	Textdateien und Datenbanken.....	342
4.5.6	Sicherung der Client-Techniken.....	342
4.5.7	PHP-Sessionfunktionen	345
4.6	Zugriff auf das Dateisystem	350
4.6.1	Einführung	350
4.6.2	Dateien lesen und schreiben	356
4.6.3	Dateienoperationen	365
4.6.4	Umgang mit Verzeichnissen	367
4.6.5	Dateiupload	372
4.7	Verbindungen zu Servern im Internet.....	378
4.7.1	Die Umgebungsvariablen des Webservers	379
4.7.2	HTTP- und FTP-Verbindungen.....	381
4.7.3	CURL-Funktionen	386
4.7.4	FTP-Funktionen	393

4.8	Sicherheit.....	400
4.8.1	Warum absichern?.....	400
4.8.2	Grundlagen der Authentifizierung.....	401
4.8.3	Seiten mit PHP schützen	406
4.8.4	Sicherheitsmethoden der Webserver	409
4.9	Bilderzeugung.....	411
4.9.1	Grundlegende Informationen.....	411
4.9.2	Prinzipielle Arbeitsweise.....	413
4.9.3	Praktischer Umgang mit Bildfunktionen	420
5	Praxis I – Lösungen für den Alltag.....	423
5.1	Lösung I: Mathematische Funktionen	425
5.1.1	Logarithmus mit unterschiedlichen Basen	425
5.1.2	Fakultät	425
5.1.3	Größter gemeinsamer Teiler	426
5.1.4	Finanzmathematische Funktionen.....	426
5.2	Lösung II: Dateioperationen	431
5.2.1	Ersetzen in Dateien.....	431
5.2.2	Umgang mit Verzeichnissen.....	432
5.2.3	Universeller Dateiupload	433
5.3	Lösung III: Counter	439
5.3.1	Textbasierter Counter	439
5.3.2	Bildbasierter Counter	440
5.4	Lösung IV: E-Mail-Abruf.....	443
5.4.1	Problemstellung.....	443
5.4.2	Code und Erläuterungen	444
5.5	Lösung V: Skripte für die Website	448
5.5.1	Minigästebuch.....	448
5.5.2	Bilderverwaltung.....	449
5.5.3	Bilderalbum mit Upload-Funktionen	450
Teil II - Datenbankprogrammierung		453
6	Grundlagen der Datenbanktechnik.....	455
6.1	Installation von MySQL.....	457
6.1.1	Installation unter Windows	457
6.1.2	Der MySQL-Monitor	458

6.2	Standardwerkzeuge für Windows	459
6.2.1	winMySQLAdmin	460
6.2.2	MySQLManager	461
6.3	Der MySQL-Dialekt	463
6.3.1	MySQL versus Standard-SQL	463
6.3.2	Erweiterungen gegenüber ANSI SQL92	463
6.3.3	Fehlende Funktionen	465
6.3.4	Ausblick auf MySQL 4	465
6.3.5	Hinweise für die Portierbarkeit	466
6.3.6	Technische Daten	467
6.4	Einführung in SQL	468
6.4.1	Grundsätzliche Konzepte	468
6.4.2	Theorie der SQL-Sprache	474
6.4.3	Literale	475
6.4.4	Datentypen	478
6.4.5	Datenbanken und Tabellen erzeugen	482
6.4.6	Mit den Daten arbeiten	487
6.4.7	Auswahlbedingungen für Abfragen	491
6.4.8	Operatoren und Funktionen	497
6.4.9	Erweiterte SQL-Programmierung	507
6.5	MySQL-Sicherheit	514
6.5.1	Einführung	514
6.5.2	Die Usertabellen in MySQL	517
6.6	ODBC	518
6.6.1	Einführung in ODBC	518
6.6.2	ODBC unter Windows einrichten	521
6.6.3	Datenbankzugriff mit Access	524
6.6.4	MySQL und ODBC	527
6.6.5	ODBC-Funktionen	528
7	Datenbankprogrammierung	529
7.1	MySQL-Funktionen	531
7.1.1	Funktionsübersicht	531
7.1.2	Erste Schritte mit MySQL und PHP	533
7.1.3	Wichtige Funktionen	535
7.1.4	Persistente Verbindungen	544

7.2	Umgang mit Datenbanken	545
7.2.1	Datenbankcursor	545
7.2.2	Eigenschaften der Ergebnisliste.....	546
7.2.3	Die Ergebnisliste auslesen	546
7.2.4	Informationen über die Datenbank.....	548
7.3	Ein Datenbankprojekt entsteht	552
7.3.1	Vorbereitung	553
7.3.2	Die Stammfunktionen.....	558
7.3.3	Realisierung.....	565
7.4	Zugriff auf ODBC-Quellen.....	570
7.4.1	Grundlagen.....	570
7.4.2	Beispielapplikation.....	573
7.5	MySQL über Access bedienen	582
7.5.1	Einrichtung	582
7.5.2	Nutzung	583
7.6	Administration mit phpMyAdmin	586
7.6.1	Einführung	586
7.6.2	Funktionsübersicht.....	590
8	Praxis II – Datenbanklösungen.....	597
8.1	Kleine Dienstprogramme	599
8.1.1	Übersicht.....	599
8.1.2	Tabellen anzeigen.....	599
8.1.3	Datumsbehandlung.....	601
8.1.4	Eine SELECT-Liste aus MySQL erstellen	603
8.1.5	Kreditkartencheck	604
8.1.6	Generierung einer WHERE-Bedingung	607
8.2	Gästebuch.....	609
8.2.1	Datenbankstruktur	609
8.2.2	Der Code des Gästebuches.....	610
8.2.3	Beschreibung	612
8.3	Authentifizierung mit Datenbank.....	612
8.3.1	Datenbankstruktur	612
8.3.2	Funktionsweise und Code.....	613
8.4	Umfrage.....	614
8.4.1	Funktionsweise	614

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

8.4.2	Datenbankstruktur	614
8.4.3	Der Code der Umfrage	615
8.4.4	Erläuterung der Arbeitsweise	621

Teil III – PHP professionell programmieren **627**

9 Fortgeschrittene Programmierung **629**

9.1	Netzwerkprogrammierung	631
9.1.1	Netzwerkfunktionen	631
9.2	E-Mail-Programmierung.....	635
9.2.1	POP3 und SMTP programmieren	636
9.2.2	E-Mail unter Windows NT/2000	646
9.2.3	E-Mail aus Skripten versenden.....	649
9.2.4	NNTP – für Newsserver programmieren	650
9.3	Portable Document Format – PDF.....	654
9.3.1	Grundlagen PDF	655
9.3.2	Einführung in die pdflib.....	655
9.3.3	PDF-Funktionen praktisch verwenden	660
9.4	Ausgabesteuerung	666
9.4.1	Grundlagen der Pufferung.....	666
9.4.2	Umgang mit den Pufferfunktionen.....	666
9.5	Reguläre Ausdrücke	668
9.5.1	Einführung	669
9.5.2	Unterschiede zu Perl	669
9.5.3	Übersicht	671
9.5.4	Die Suchmuster im Detail.....	671
9.5.5	Die PCRE-Funktionen im Detail.....	692
9.5.6	Ersetzungen mit regulären Ausdrücken	694
9.6	Extensible Markup Language – XML	697
9.6.1	Was ist XML?.....	697
9.6.2	Auswahl des XML-Parsers	700
9.6.3	Die Funktionsdeklarationen.....	701
9.6.4	Funktionsübersicht XML-Funktionen	703
9.6.5	XML-Anwendungen in der Praxis	704
9.6.6	Das XML-Projekt.....	711
9.7	Wireless Markup Language – WML	724

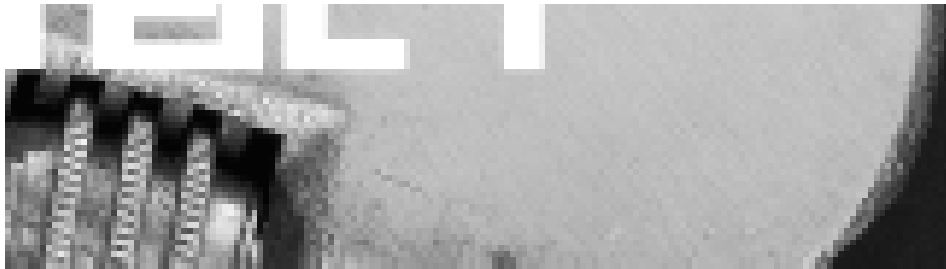
9.7.1	Grundlagen WML	724
9.7.2	Praxis: WAP-Telefondatenbank	726
9.7.3	Praktischer Einsatz von WAP/WML	732
9.8	Web Distributed Data Exchange – WDDX	733
9.8.1	WDDX im Detail.....	733
9.8.2	WDDX-Elemente	737
9.8.3	WDDX-Funktionen in PHP	741
10	Zusatzprogramme	747
10.1	Editoren	749
10.1.1	Editoren für Windows	749
10.1.2	Editoren für Unix.....	750
10.2	ZendIDE	750
10.2.1	Vorbereitung	751
10.2.2	Installation	751
10.2.3	Grundfunktionen.....	753
10.2.4	Die Fenster der IDE	756
10.2.5	Verwendung der Debuggerfunktionen.....	758
10.3	Zend-Hilfsprogramme.....	762
10.3.1	Zend Optimizer	762
10.3.2	Zend Cache.....	764
10.3.3	Zend Encoder Unlimited	764
10.3.4	Zend Launch Pad.....	765
11	Praxis III – Das PHP-Projekt	767
11.1	phpTemple – Die Idee	769
11.1.1	Templates: Trennung von Code und Design	769
11.1.2	Templatesysteme für PHP.....	770
11.1.3	Die Idee: phpTemple.....	772
11.2	Definition des Leistungsumfanges.....	772
11.2.1	Grundsätzlicher Aufbau.....	773
11.2.2	Variablen in Vorlagen	775
11.2.3	Das Sitemanagement.....	777
11.2.4	Das Formularmanagement.....	777
11.2.5	Zugriff auf Daten aus Datenbanken	785
11.2.6	Steuerung in der Vorlage.....	792
11.2.7	Zugriff auf PHP	794

11.2.8	HTML-spezifische Funktionen	795
11.3	Installation und Administration	800
11.3.1	Aufbau der Konfigurationsdatei	801
11.3.2	Administration	802
11.4	Grundlegende Hinweise zum Prinzip	804
11.4.1	Prinzip der Linkauflösung	804
11.4.2	Einschränkungen	805
11.5	Anwendungsbeispiele	805
11.5.1	Eine Navigation aufbauen	805
11.5.2	Grafiken dynamisch erzeugen	806
11.6	Realisierung: So funktioniert es!	808
11.6.1	Startdatei	808
11.6.2	Sessionverwaltung	810
11.6.3	Das Kernstück: der Parser	816
11.6.4	Formularsteuerung	835
11.6.5	Datenbankzugriff	839
11.6.6	Installation und Administration	840
11.7	phpTemple-Projekte im Web	840
11.7.1	Mauerfotos	841
11.7.2	Weitere Projekte	844
11.8	phpTemple für professionellen Einsatz	845
11.8.1	Lizenzrechtliche Fragen	845
11.8.2	Programmtechnische Modifikationen	845
11.8.3	Weitere Eigenschaften	846
11.8.4	Kontaktadresse	846

Teil IV – Anhänge und Referenz	847
A Glossar	849
B Kurzreferenz	883
B.1 Vorbemerkungen	883
B.2 Alphabetische Übersicht	884
B.3 Befehlsreferenz	907
B.4 Funktionsreferenz	912
B.5 Datenbankfunktionen	1032
B.6 Systemnahe Funktionen	1068
B.7 MySQL-Referenz	1080
C Server-Variablen und Statuscodes	1093
C.1 Server-Variablen	1093
C.2 HTTP-Statuscodes	1097
C.3 Die Codes im Detail	1098
D Index	1105
D.1 Erläuterungen zum Index	1105
D.2 Globaler Index	1106
E An den Autor	1125

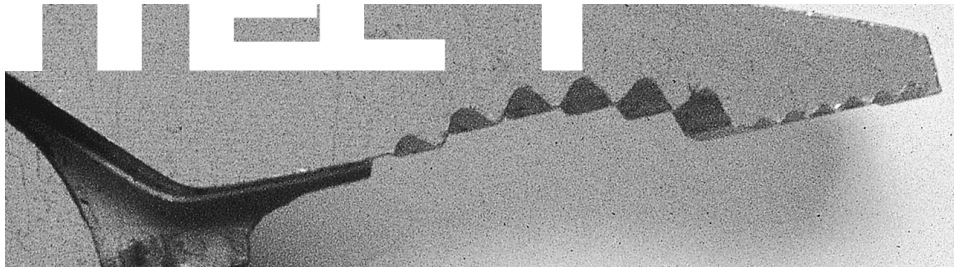
V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

TEIL I



Einführung und Grundlagen

1



Einführung

Dieses Kapitel enthält einen kurzen geschichtlichen Abriss zu PHP, eine Definition der Zielgruppe und Hinweise zum Umgang mit dem Buch, der CD und zur Website zu diesem Buch.

Ein Ausblick auf die wichtigsten Betriebssysteme Linux und Windows schließt sich an. Anfänger finden Tipps zum sauberen Programmieren.

Was ist PHP? (Seite 25)

Zielgruppe (Seite 27)

Wie dieses Buch zu lesen ist (Seite 28)

Die Buch-CD und Website (Seite 32)

Dynamische Webseiten (Seite 36)

Linux oder Windows? (Seite 37)

Was ist Open Source? (Seite 42)

Tipps zum sauberen Programmieren (Seite 43)

1.1 Was ist PHP?

PHP ist eine Skriptsprache zur Erstellung dynamischer Websites, nicht mehr, aber auch nicht weniger. Die erste Version entwickelte 1994 Rasmus Lerdorf, der eigentlich nur eine Möglichkeit zur Programmierung seines eigenen Webserver suchte. Er nannte seine kleine Skriptmaschine »Personal Home Page Tools«. Die Applikation stellte er ins Netz und ließ die freie Verbreitung zu. So entstand PHP, als Abkürzung zu »Personal Home Page«. Später entwickelten die Open Source-Jünger, bekannt für kryptische Akronyme, die rekursive Version »PHP HyperText Preprocessor«. Einen offiziellen Namen gibt es darüber hinaus nicht.

PHP ist eine Skriptsprache

Den Weg zum professionellen Programm ging PHP erst 1995. Unter dem Namen PHP/FI (Personal Homepage Tools/Form Interface) wurde die erste wirklich nutzbare Version veröffentlicht. PHP/FI wird als Version 2 angesehen. PHP entwickelte sich seit dieser Zeit rasant weiter. Zum einen trug dazu die freie Verfügbarkeit bei. Die ist bei ASP und bei Perl aber auch gegeben. PHP ist im Gegensatz dazu eine Sprache, deren einziger Zweck das Web ist. ASP kommt mit VBScript daher, einem BASIC-Derivat, das nur schwer an die Bedürfnisse eines Webserver angepasst werden kann. Moderne Sprachelemente wie die objektorientierte Programmierung sind nur unvollkommen umgesetzt. Der größte Nachteil ist aber die Einschränkung auf Microsoft-Plattformen. Perl wurde schon 1986 von Larry Wall entworfen, zu dieser Zeit war das Web noch unbekannt. Die starken Zeichenkettenfunktionen und die freie Verbreitung als Open Source machten Perl den Weg ins Web frei. Leider ist die Lesbarkeit nicht einfach und die Programmierung vor allem für Anfänger sehr kompliziert.

1995: PHP/FI

Aus der Beobachtung der Nachteile von Perl entstand eine Skriptsprache, die diese vermeidet und dafür alle Vorteile der Konkurrenten in sich vereint – PHP. Zum einen wird die von ASP bekannte einfache Kombination mit HTML genutzt, zum anderen die leistungsstarke und verbreitete C-Syntax, die vielen schon von JavaScript oder Java bekannt ist. Objektorientiertes Programmieren ist ansatzweise möglich, ebenso wie eine gute Modularisierung.

Vorteile

PHP besticht außerdem durch einen fast schon grandiosen Funktionsumfang. Hier kann keine andere Sprache mithalten. Wo immer ein Programmierer ein Problem zu lösen hatte, wurde der Funktionsumfang erweitert. Besonders deutlich wird dies bei den Datenbankfunktionen. PHP unterstützt mehr Datenbanken direkt, als die meisten Programmierer überhaupt vom Namen her kennen. Den Erfolg macht nicht zuletzt auch die gute Unterstützung für die Datenbank MySQL aus, die wie PHP auch als Open Source verfügbar ist.

- 1997: PHP 3** Seit Mitte 1997 wird PHP von einem Team an Programmierern rund um Rasmus Lerdorf weiterentwickelt. Mit der Version 3 stand bereits Anfang 1999 ein System zur Verfügung, das keine Konkurrenz seitens Perl oder ASP mehr fürchten muss.
- 2000: PHP 4 / Zend** Der nächste Schritt, PHP 4, wurde im Jahr 2000 vollendet. Dieses Buch geht vollständig auf PHP 4 ein. Im Zusammenhang mit PHP 4 ist oft auch von Zend die Rede. Zend ist ein Kunstwort aus »Zeev« und »aNDi«, den Vornamen der Entwickler Zeev Suraski und Andi Gutmán. Zend ist aber keine Konkurrenz zu PHP, sondern der Kernel der neuen Version PHP 4. Zend 1.0 sorgt dafür, dass PHP 4 deutlich schneller und leistungsfähiger als PHP 3 ist. Zusätzlich gibt es einen »Zend Optimizer«, der die Leistung der Skripte nochmals verbessert.
- 2001: Jahr der Diskussionen** Eigentlich nichts neues möchte man meinen. Die PHP 4-Entwickler ruhen sich jedoch nicht auf ihren Lorbeeren aus, sondern bringen im Abstand weniger Wochen neue Versionen heraus. Auch wenn die Versionsnummer erst in der dritten Stelle wechseln – zum Erscheinungstermin dieses Buches erschien PHP 4.0.7 – zeigen sich dennoch Fortschritte. So werden immer neue Bibliotheken integriert, was zu einem drastischen Anwachsen der Funktionsanzahl führt. Zum anderen wird PHP immer stabiler und schneller. Diskutiert wird auch zunehmend – über die Zukunft von PHP, über PHP 5 und verschiedene Aktivitäten der Entwickler. Kritisch werden kommerzielle Zusatzprodukte der Zend-Entwickler gesehen, darunter der Zend-Encoder (verschlüsselt PHP-Code) und der Zend-Cache (beschleunigt die Abarbeitung). Beides Programme, die nicht mehr Open Source sind, sondern gegen einen nicht unerheblichen Betrag verkauft werden. Natürlich reagierte die Szene hier und es gibt alternative Projekte unter der GPL (*General Public License*). Damit entzündete sich natürlich die Diskussion, wer künftig über den Kern der Sprache mitbestimmen darf. Auch die Forderungen nach einem Applikationsserver – ähnlich Java – werden immer wieder erhoben. Auch hier ist von Zend nichts zu sehen. Einige lehnen derart komplexe Programme ab, weil es den Einsatz erschwert und damit die massenweise Verwendung von PHP behindert. Professionelle Entwickler spüren jedoch die Grenzen, die ihnen PHP setzt und fühlen sich – je nach Systemwelt – zu Microsofts .Net oder Suns Java getrieben. Wie es weiter geht, wird sicher erst 2002 entschieden werden. Bis dahin bleibt dieses Buch ihr treuer und kompetenter Begleiter.



In diesem Buch ist von PHP 3 und PHP 4 die Rede, Unterschiede werden klar herausgestellt. Erfolgt keine Unterscheidung, sind beide Versionen in diesem Punkt kompatibel. Grundsätzlich bildet PHP 4 aber den Schwerpunkt, die beiliegenden Skripte wurden nicht alle mit PHP 3 getestet.

1.2 Zielgruppe

Sie programmieren HTML-Seiten? Sie möchten Ihren Webserver selbst programmieren? Sie möchten Ihre Seiten mit Datenbanken aufwerten? Dann sind Sie die Zielgruppe.

PHP ist eine umfangreiche, leistungsstarke Skriptsprache zur Programmierung dynamischer, datenbankgestützter Websites. Das klingt komplizierter, als es ist. Mit Hilfe dieses Buches sollten auch Webdesigner, die bisher keine Erfahrung mit der Programmierung hatten, in die Welt der Webserverprogrammierung eintauchen können. Ich habe das Buch so aufgebaut, dass in den ersten Abschnitten keine Kenntnisse – abgesehen von HTML – vorausgesetzt werden. Profis, die andere Programmiersprachen kennen, sei die Cross-Referenz im Anhang ans Herz gelegt. Dort finden Sie eine Gegenüberstellung gängiger Befehle aus Perl, JavaScript und VBScript mit den Entsprechungen in PHP. Mit dieser Hilfe können Sie Skripte leicht übertragen.

Betriebssystemkenntnisse sind ebenfalls vorauszusetzen, die Plattform ist bei PHP fast freigestellt. Da der PHP-Prozessor als C-Quelltext vorliegt, kann er leicht auf alle Plattformen übersetzt werden. Entsprechend groß ist die Auswahl an Binärdistributionen. Wenn Sie mit Windows vertraut sind, sollten Sie PHP unbedingt auch unter Windows entwickeln. Haben Sie mehr für Linux übrig, nutzen Sie dies auch als Entwicklungsplattform. Das gilt gleichermaßen für alle anderen Systeme. Dieses Buch entstand in weiten Teilen auf einem Windows-System (Windows 2000), was aber mehr an der Verfügbarkeit eines geeigneten Satzprogramms lag. Die Hauptplattform dürfte Linux sein, an entsprechenden Stellen wird auf Besonderheiten eingegangen.

Wer PHP bereits gut kennt und Informationen für Hardcore-Programmierer erwartet, wird etwas enttäuscht sein. Zum einen bietet das Buch trotz beachtlichen Umfangs nicht unbegrenzt Platz. Zum anderen sind es nur sehr wenige Personen, die Bedarf an einem sehr tiefen Einstieg in PHP haben. Die benötigten Informationen sind dann aber nur noch schwer zu vermitteln und kaum auf engem Raum darstellbar. Die Funktionserweiterung durch Eingriffe in den Quelltext wird daher nur am Rande angesprochen. Bedenken Sie auch, dass sich PHP rasant weiterentwickelt und Änderungen am PHP 3-Code sinnlos werden, wenn PHP 4 verwendet wird. Der Aufwand zur langfristigen Pflege solcher Lösungen ist enorm. Versuchen Sie daher, komplexe Probleme zuerst mit PHP-eigenen Mitteln zu lösen. In diesem Sinne sind viele Beispiele explizit auf PHP 4 abgestimmt.

Noch eine andere Zielgruppe wird sich zwangsläufig mit PHP beschäftigen müssen – die Internet Service Provider (ISP). Immer mehr Kunden, die Webspace mieten, werden eine Unterstützung für Skripte

Für wen ist dieses Buch?

**Betriebssystem-
kenntnisse**

Was es nicht gibt

Für ISPs

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

erwarten. Die Verbreitung von PHP führt dazu, dass die Forderungen direkt an die ISPs herangetragen werden. Da PHP sehr umfangreiche Zugriffe auf den Server erlaubt, sollten Sie mit den Möglichkeiten vertraut sein. Sie könnten sonst Sicherheitslöcher groß wie Scheunentore auf tun.

1.3 Wie dieses Buch zu lesen ist

Sicher bleibt es jedem Leser überlassen, wie er das Buch liest. Für die ersten Schritte ist es aber effektiver, die Struktur ungefähr zu kennen.

1.3.1 Aufteilung

Das Buch ist in vier Teile gegliedert, die sich jeweils einer bestimmten Zielgruppe widmen:

Teil I – Einführung und Grundlagen

Dieser Teil ist für alle Anfänger und Einsteiger interessant. Es werden einige theoretische Grundlagen vermittelt, die später noch vertieft werden. Außerdem wird auf die Installation eingegangen. Die Erklärung der Basisfunktionen und -befehle finden Sie auch in diesem Teil.

Teil II – Datenbankprogrammierung

Hier geht es vor allem um Datenbanken, speziell MySQL. Sie erfahren alles über SQL, ODBC und theoretische Grundlagen zu Datenbanken. Die praktische Arbeit mit PHP und MySQL steht im Vordergrund dieses Teils.

Teil III – PHP professionell programmieren

Dieser Teil wendet sich an Leser, die bereits erste Erfahrungen mit PHP gesammelt haben und nun mehr aus ihrem System herausholen wollen. Vertiefend wird auf Protokolle eingegangen und umfangreiches Hintergrundwissen vermittelt. Jede Funktion wird auch in diesem Teil mit Beispielen erklärt. Ein umfangreiches Profi-Projekt zeigt, was Sie mit PHP erreichen können.

Teil IV – Anhänge und Referenz

Mit diesem Teil arbeiten Sie, wenn Sie PHP beherrschen und eine schnelle Übersicht zum Nachschlagen benötigen. Sie finden eine Kurzreferenz und eine MySQL-Befehlsübersicht.

1.3.2 Kapitelübersicht

Die folgende Übersicht zeigt alle Kapitel auf einen Blick, damit Sie sich **Kapitelübersicht** schnell grob orientieren können.

1 Einführung	23
2 Vorbereitung und Installation	45
3 Erste Schritte mit PHP	145
4 Interaktive Webseiten	299
5 Praxis I – Lösungen für den Alltag	423
6 Grundlagen der Datenbanktechnik	455
7 Datenbankprogrammierung	529
8 Praxis II – Datenbanklösungen	597
9 Fortgeschrittene Programmierung	629
10 Zusatzprogramme	747
11 Praxis III – Das PHP-Projekt	767

Am Anfang eines jeden Kapitels finden Sie eine Übersicht über die Abschnitte und deren Inhalt, jeweils mit eigenen Verweisen auf die entsprechenden Seiten. Diese zusätzliche Navigationsebene umgeht bewusst das umfangreiche Inhaltsverzeichnis und soll vor allem am Anfang die Orientierung erleichtern. **Übersichten am Kapitelanfang**

1.3.3 Icons und Symbole

Im Buch werden an vielen Stellen in der Marginalspalte Symbole verwendet. Sie dienen der schnellen Orientierung beim Durchblättern. Da viele Informationen an mehreren Stellen zu finden sind, dienen bestimmte Symbole der besseren Erkennbarkeit der Relevanz einer Textstelle. Auf diese Weise kann der Index etwas entlastet werden, der bewusst nur auf die Schwerpunktinformationen hinweist.

Dieses Symbol bezeichnet Probleme, Bugs, ernste Hinweise auf Gefahren oder auch besonders wichtige Informationen. Warnungen stehen außerdem immer in einem grauen Kasten.



Warnung

Tipps sind besonders gekennzeichnet, um die Aufmerksamkeit auf eine bestimmte Stelle im Text zu lenken. Dieses Symbol finden Sie vor allem bei langen Texten zur Strukturierung.



Tipp



Hinweis

Hinweise kennzeichnen interessante Ergänzungen zum Text, die den unmittelbaren Kontext betreffen oder ergänzen. Ein grauer Kasten hebt die Bedeutung zusätzlich hervor.



Das CD-Symbol finden Sie, wenn die Skripte auf der beiliegenden CD zu finden sind, jedoch nicht vollständig abgedruckt wurden. Alle Programme auf CD stehen auch auf der Website zur Verfügung, die Sie unbedingt besuchen sollten, um die aktuellsten Versionen verwenden zu können.



Beispiel

Einige Skripte sind nicht auf der CD, sondern dienen nur der Erläuterung der Anwendung. Oder es gibt im Internet eine Quelle, unter der Sie die aktuellste Version beziehen können. Diese Skripte sind mit dem Beispiel-Symbol gekennzeichnet.



Übung

Dieses Symbol weist Sie auf Übungen hin. Sie können Ihre Fähigkeiten, PHP zu programmieren, an diesen Stellen besonders ausbauen. Zum Verständnis des Buches sind diese »Lehrabschnitte« aber nicht notwendig. Da dies kein Lehrbuch im klassischen Sinne ist, sind derartige Übungen sehr selten.



Theorie

An vielen Stellen im Buch finden Sie theoretische Basisinformationen, die zum unmittelbaren Verständnis nicht notwendig sind. Diese Grundlagen sind vor allem für Leser gedacht, die gleichzeitig Hintergrundwissen erwerben möchten, beispielsweise für die berufliche Fortbildung.



Insider

Insider-Informationen haben eher einen praktischen Bezug, gehen jedoch ebenso über das zur reinen Programmierung nötige Niveau hinaus. Auch »geheimnisumwitterte« Hintergrundinformationen werden so gekennzeichnet.



Windows

PHP ist eine Skriptsprache für die plattformunabhängige Programmierung. Trotz aller Bemühungen der Entwickler gibt es einige Unterschiede zwischen Windows und Unix. Abschnitte, die speziell zu Windows Informationen bieten, sind mit dem Windows-Icon gekennzeichnet. Ohne besondere Kennzeichnung beziehen sich die Angaben immer auf alle Plattformen.



Immer noch laufen weltweit PHP-Server unter PHP 3. Die neue Version PHP 4 realisiert ab Mitte 2000 durch die leistungsstarke Zend-Engine eine neue Dimension der Skriptsprache. Dieses Buch geht im Wesentlichen auf PHP 4 ein. In weiten Teilen ist PHP 4 kompatibel zu PHP 3. Wenn Abschnitte *ausschließlich* für PHP 4 gelten, wird dies mit dem PHP 4-Logo gekennzeichnet.

1.3.4 Schreibweisen und Satz

Im Buch werden konsequent bestimmte Schreibweisen genutzt. Davon sollen vor allem Anfänger profitieren, die nicht auf Anhieb unter-

scheiden können, ob es sich bei einem Namen um ein selbstgewähltes Sprachkonstrukt oder einen reservierten Bezeichner handelt.

Befehle, Namen, Konstanten, Bezeichner usw., die in PHP fest definiert sind, werden mit einer nichtproportionalen Schrift gesetzt und sehen dann folgendermaßen aus: `echo`, `function`, `class`. **Reservierte Wörter**

Wenn dagegen Namen frei definiert wurden, beispielsweise die Namen von nutzerdefinierten Funktionen oder Variablen, werden diese im Fließtext kursiv gesetzt: *myfunction*, *var3*. **Eigene Namen**

Kursiv sind auch alle Links (URLs) gesetzt, die Sie im Internet aufsuchen können und die zu einem Thema weitergehende Informationen bieten, beispielsweise *<http://www.php.net>*. **URLs**

Quellcode, den Sie eintippen oder von der CD laden und ausführen können, wird in einer nichtproportionalen Schrift gesetzt: **Quellcode**

```
<HTML>
<HEAD>
<TITLE>Testprogramm</TITLE>
</HEAD>
<BODY>
<H1>Testprogramm</H1>
</BODY>
</HTML>
```

Oft werden längere Quelltexte abgedruckt, um das Programm im Überblick zu zeigen. Die wichtigsten oder kompliziertesten Passagen werden anschließend noch einmal erläutert. Solche Fragmente können Sie oft nicht einzeln eingeben und ausführen, sie sind deshalb grau hinterlegt:

```
<HEAD>
<TITLE>Testprogramm</TITLE>
</HEAD>
```

Oft werden auch Eingaben an der Konsole notwendig. Dann werden diese wie Quellcode in einer nichtproportionalen Schrift fett gesetzt. Ausgaben erscheinen dagegen normal: **Eingaben**

```
$ mount c:/temp /tmp
Successfully mounted device tmp
```

Dialogfelder, Menübefehle oder andere vorgegebene Elemente werden in Kapitälchen gesetzt. Aufeinanderfolgende Menübefehle werden durch einen senkrechten Strich getrennt: DATEI | ÖFFNEN. **Dialoge und Menüs**

Wenn es sich um fest vergebene Dateinamen handelt, beispielsweise vom System vorgegebene, werden diese ebenfalls in Kapitälchen gesetzt: PHP.INI. **Dateinamen**

1.4 Die Buch-CD und Website

Zu diesem Buch gibt es eine CD und eine Website, auf der aktuellere Versionen der Skripte und der Software zu finden sind.

1.4.1 Allgemeines



Beachten Sie, dass diese Skripte teilweise von der abgedruckten Version abweichen. Gedruckte Skripte sind oft gekürzt, von Kommentaren befreit und für den Satz anders umbrochen. Verwenden Sie im Zweifelsfall immer die Version von CD. Alle Skripte sind auch im Internet zu finden, wo Sie auch aktuellere Versionen und Bugfixes finden. Die Buch-CD hat eine eigene Bedienoberfläche, die den Zugriff auf alle Bestandteile beinhaltet. Wenn Sie PHP-Skripte aus der CD heraus ausführen möchten, müssen Sie den Inhalt zuvor auf Ihre lokale Festplatte oder einen Webserver kopieren und den Server so einrichten, dass er mit PHP umgehen kann. Lesen Sie dazu Kapitel 2. Wenn Sie keinen Webserver haben und nur zur Information mit PHP spielen wollen, nutzen Sie die Website zum Buch:

Website zum Buch <http://www.php.comzept.de>

CD und Website sind weitestgehend identisch, die Website wird allerdings gelegentlich aktualisiert. Mehr Informationen zu PHP 4 erfahren Sie auf der Partnersite zum Buch unter:

Partnersite zum Buch <http://www.phparchiv.de>

Installation der Daten von der CD

Voraussetzungen Auf der CD finden Sie ein Programm, das den bequemen Zugriff auf die Skripte ermöglicht, sowie eine Sammlung der Skripte in fertiger Form. Das Programm nutzt XML-Funktionen und setzt einen fertig installierten Webserver und ein konfiguriertes PHP 4 voraus, was am Anfang kaum gegeben sein dürfte. Nutzen Sie deshalb zuerst die Skripte im Verzeichnis READYSCRIPT. Mehr Informationen zur Installation finden Sie, wenn Sie die Datei INDEX.HTM im Stammverzeichnis der CD aufrufen.

Struktur der CD Die CD selbst hat folgende Struktur:

- /READYSCRIPT

Fertige Skripte zu allen Listings im Buch mit der Dateierweiterung .PHP4. Entscheidend ist der Name der Skripte.

- /ANWENDUNGEN

Einige größere Programme sind hier in eigenen Verzeichnissen zu finden.

- /DATA

Hier finden Sie die für die Skripte notwendigen Testdaten.

- /XMLBASE

Dies sind die im XML-Format abgelegten Basisdaten der Skripte, die zur Generierung der Website und der CD-Daten verwendet wurden. In diesem Verzeichnis sind auch die PHP-Skripte zu finden, mit denen die Daten erzeugt wurden.

- /SOFTWARE

Hier sind alle für die Installation nötigen Programme, also PHP selbst, die aktuellen Distributionen und Quelltexte, der Apache-Webserver und MySQL zu finden (Stand September 2001). Bei Fragen zur Installation wenden sie sich an die Hersteller bzw. informieren Sie sich auf der Website der jeweiligen Anbieter und in den beiliegenden Informationsdateien.

Um die Daten von der CD interaktiv verwenden zu können, gehen Sie folgendermaßen vor:

**So verwenden Sie
die CD interaktiv**

1. Stellen Sie sicher, dass der Webserver (Apache oder IIS), PHP 4 (mindestens Version 4.0.5) und MySQL (ab Version 3.23) problemlos läuft.
2. Kopieren Sie die beiden Verzeichnisse /ANWENDUNGEN und /XMLBASE in ein neues Verzeichnis unterhalb des Stammverzeichnisses des Webserver, beispielsweise mit dem Namen /PHPBOOK.
3. Entfernen Sie dann den Schreibschutz von den Skripten und den Verzeichnissen und geben Sie Schreibrechte für das anonyme Benutzerkonto frei (einige Skripte schreiben Daten).
4. Kopieren Sie dann den Inhalt des Verzeichnisses /DATA in das neu angelegte Verzeichnis /PHPBOOK.
5. Legen Sie zwei neue Verzeichnisse mit den Namen /UPLOAD und /LOGDATA an (diese werden von einigen Skripten benötigt). Achten Sie darauf, dass auch hier Schreibrechte bestehen müssen.
6. Richten Sie MySQL ein und führen Sie die Anweisungen in der Datei INSTALL.SQL aus, am Besten mit einem Werkzeug wie *phpMyAdmin*. Dann passen Sie die Datei OPEN.INC.PHP4 so an, dass die Daten zu Ihrem Datenbankserver passen.
7. Starten Sie die Datei INDEX.PHP4 im neu angelegten Verzeichnis /PHPBOOK.

Sie sollten jetzt die Übersicht der Kapitel sehen und die Listen der Skripte mit Namen und Listingnummern. Da die Informationen aus den XML-Dateien extrahiert werden, kann dies einen Moment dauern.

Tipps für Anfänger

Wenn Sie noch keine Erfahrung mit PHP haben und erst lernen wollen, verwenden Sie die fertigen Skripte aus dem Verzeichnis /READYSCRIPT. Kopieren Sie diese in ein Verzeichnis des Webserver und rufen Sie sie dann anhand des Namens auf. Wenn Sie noch keinen Webserver haben, können Sie alle Skripte auch über die Website zum Buch ausprobieren – freilich ohne die Möglichkeit, den Code zu verändern. Die Website erreichen Sie unter folgender Adresse:

```
http://www.php.comzept.de
```

1.4.2 Quellen im Internet

Im Internet finden Sie immer aktuellere Informationen als in einem Buch, das nur gelegentlich neu aufgelegt werden kann.

Dokumentationen

PHP ist eine sehr lebendige und zügig entwickelte Sprache. Dieses Buch berücksichtigt die neueste Version PHP 4 auf Basis der Zend-Engine. Bei Drucklegung standen nicht für alle Funktionen Originaldokumentationen zur Verfügung. Der Autor war deshalb an einigen Stellen auf eigene Recherchen und Experimente angewiesen. Darüber hinaus basiert dieses Buch auf der Version PHP 4.0.6. Sie sollten sich daher auf den folgenden Servern regelmäßig über den Fortgang der Arbeiten informieren:

```
http://www.php.net
```

```
http://www.zend.com
```

Wenn Sie eine neuere oder ältere Version einsetzen, gibt es in seltenen Fällen Unterschiede bei einigen Funktionen.

Tutorien

Wenn Sie statt der systematischen Erlernung nach schnellen Problemlösungen suchen und im Buch nicht genau die Antwort finden, helfen Tutorien oft weiter. Dies sind kompakte Lehrgänge, die sich in wenigen Minuten absolvieren lassen und auf ein spezielles Thema eingehen. Beispiele finden Sie unter den folgenden Adressen im Internet:

```
http://www.dynamic-webpages.de/05.tutorials.php
```

```
http://www.php-center.de/tutorial
```

Beispielskripte

Beispielskripte stehen in der Regel frei zur Verfügung, einige Skripte in diesem Buch wurden den folgenden Seiten entnommen, die eine unerschöpfliche Quelle cleverer Lösungen darstellen:

<http://px.sklar.com>

<http://www.weberdev.com>

<http://www.phparchiv.de>

Mailinglisten

Eine sehr effektive Informationsquelle sind Mailinglisten. Die aktivste Mailingliste erreichen Sie unter *php-general@list.php.net*. Die Liste ist allerdings extrem »noisy«, das heißt, die Anzahl an brauchbaren und auch unbrauchbaren Postings ist sehr hoch. An Spitzentagen können Sie schon mal mit einigen Hundert E-Mails rechnen. Glücklicherweise gibt es ein Archiv, in dem Sie auch die aktuellsten Postings finden. Die Webadresse ist *http://www.php.net/list*. Und Freunde des Usenet finden die passende, inhaltsgleiche Newsgroup unter *nnntp://news.php.net*. Allerdings können Sie an diese Newsgroup nicht posten. Sie müssen Fragen an die Mailingliste schicken und die Antworten erscheinen dann, zusammen mit der Frage, auch in der Newsgroup. Wenn Sie mit der Liste arbeiten möchten, bietet sich der Digest-Dienst an, der zweimal am Tag die »gesammelten Werke« als kompakte E-Mail versendet. Vor allem Anfängern, die nur als Lurker¹ an der Liste teilnehmen, kann ich diese Form empfehlen.

Weitere Informationen über internationale Listen zu PHP finden Sie unter folgender Adresse:

<http://www.php.net/support.php>

Newsforum

Es existiert neben dem schon erwähnten »Read-Only«-Forum noch eine echte deutsche Newsgroup mit dem Namen *DE.COMP.LANG.PHP*. Wenn Ihr Provider die Gruppe nicht führt, schauen Sie unter folgender Adresse nach:

<http://www.deja.com>

Mailinglisten bieten oft »Erste Hilfe« bei Problemen

Im Usenet

¹ Lurker (vom engl. Lurk = Versteck) sind Personen, die in öffentlichen Listen nur als Leser teilnehmen. Lurken wird Anfängern empfohlen, um die Gewohnheiten einer Liste oder Newsgroup kennen zu lernen und zu vermeiden, dass zu viele Personen zu simple Fragen stellen.

IRC – Internet Relay Chat

Im Efnet

Es gibt einen internationalen und einen deutschen Channel im IRC, konkret im Efnet. Die Qualität schwankt sehr stark zwischen hervorragenden Beiträgen der Entwickler und völligem OT². Die Channel heißen #PHP und #PHPNET.

Foren

Im WWW

Reine Foren sind praktisch webbasierte Listen. Der Umgang ist einfacher und weniger arbeitsintensiv als bei Mailinglisten, verlangt aber etwas Eigeninitiative. Wählen Sie beispielsweise die folgenden Adressen:

<http://analogon.com/php/forum>

<http://www.phpwelt.de/powerboard/>

1.5 Dynamische Webseiten

Grundlagenwissen



Unter dynamischen Webseiten werden Seiten verstanden, deren endgültige, an den Server gesendete Form erst im Augenblick des Abrufes entsteht. So können Daten interaktiv in die Seiten eingebaut werden.

Der Vorteil besteht vor allem in der Möglichkeit, auf Nutzereingaben reagieren zu können. Formulare lassen sich sofort auswerten und schon die nächste Seite kann den Inhalt wiedergeben oder Reaktionen darauf zeigen. Die Anwendungsmöglichkeiten sind fast unbegrenzt. Ob und in welchem Umfang außerdem Datenbanken zum Einsatz kommen, hängt von der Zielstellung ab. Dynamische Webseiten an sich benötigen keine Datenbank. Sie sollten sich vor allem als Anfänger nicht dem Zwang unterziehen, gleich jedes Problem mit der Hilfe einer Datenbank zu lösen, auch wenn Profis dies bevorzugen würden. Im Buch werden viele Beispiele gezeigt, die mit einfachsten Mitteln beeindruckende Effekte erzielen – ganz ohne Datenbank.

Die Entstehung einer dynamischen Website wird in Abbildung 1.1 erläutert. Zuerst fordert der Nutzer mit seinem Browser ein Skript an. Der Webserver leitet diese Anfrage aufgrund der Dateierweiterung an ein bestimmtes Programm weiter, beispielsweise das PHP-Modul. Dort wird die Seite durchsucht und darin enthaltene Codes werden ausgeführt. Daraus entsteht wiederum HTML-Code, einschließlich der Daten aus Datenbankabfragen oder früheren Nutzereingaben. Die fertige Seite wird dem Webserver zurückgegeben, der sie dann an den Browser sendet.

² In Newsgroups und Listen werden Beiträge, die nicht zum Hauptthema passen, als »Off Topic« = OT bezeichnet.

Für das Skript besteht damit jede Freiheit, den HTML-Code zu erzeugen und mit Daten anzureichern. Als Skriptsprache kommen viele Möglichkeiten in Betracht, von denen PHP sicher eine der attraktivsten ist. Andere Skriptumgebungen sind ASP (Active Server Pages) mit den Sprachen VBScript und JScript, ASP+ mit zusätzlich C# (sprich »C Sharp«) und natürlich Perl. Seltener verwendet werden JavaScript (Netscape Webserver) oder Python.

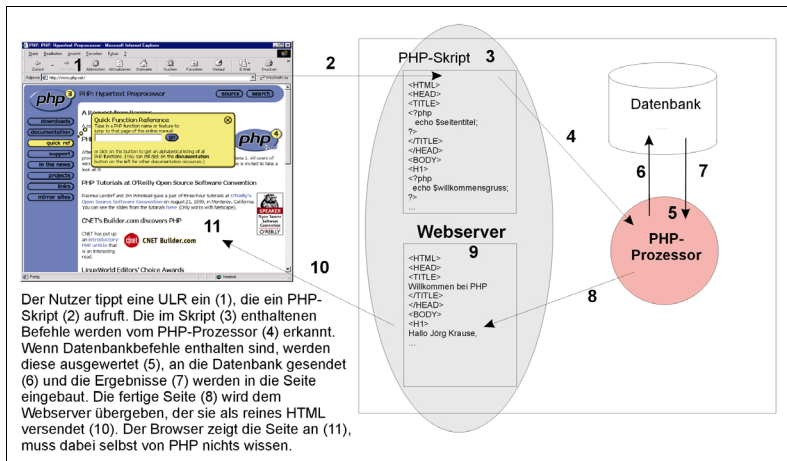


Abbildung 1.1:
So entsteht eine
dynamische Website

1.6 Linux oder Windows?

Die Frage nach dem richtigen Betriebssystem wird oft gestellt. Da PHP aus der Open Source-Szene kommt, ist Linux klar favorisiert. Wenn Sie noch keine feste Beziehung zu einem Betriebssystem haben, ist die Wahl nicht ganz so einfach.

Gewissensfrage Linux

Linux als alternative Plattform zu Microsofts Windows ist in aller Munde. Kaum eine Zeitschrift ohne Artikel, kaum ein Leserbrief ohne begeisterten Kommentar. Aber wozu stellt Linux wirklich eine Alternative dar? Im Web tobt ein regelrechter Glaubenskrieg zwischen den Anhängern von Linux und Windows. PHP kommt aus der Unix-Welt und hat sich besonders schnell auf Linux-Servern verbreitet. Wenn Sie noch vor der Wahl der Plattform stehen, sollten Sie Linux ernsthaft als primäre Variante in Betracht ziehen. Vielleicht helfen Ihnen ein paar Hintergrundinformationen weiter.

Das Betriebssystem Linux ist ein UNIX-Derivat. Im Gegensatz zu fast allen anderen Unix-Versionen ist es unter der General Public License (GPL) veröffentlicht, im Quellcode zugänglich und als Open Source (siehe nächster Abschnitt) erhältlich. Diese Form des »Softwarever-

Ist Linux eine
Alternative?

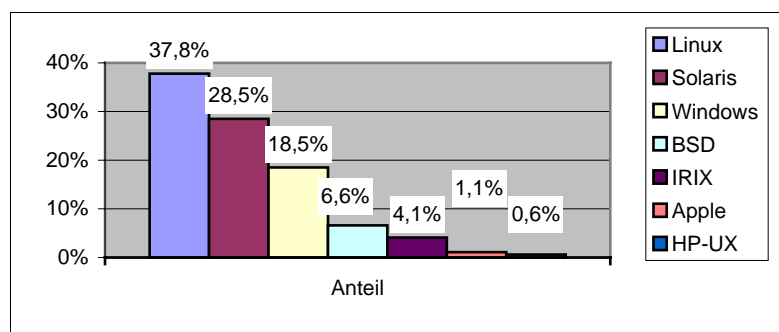
triebs« ist im kommerziell orientierten Softwaremarkt kaum bekannt und birgt einige Besonderheiten. So sind zwar Quellen und diverse Hilfsprogramme auf Webservern im Internet frei verfügbar, eine praktisch nutzbare Form stellen diese jedoch nicht dar. Linux ist ein System vor allem für Entwickler. Es arbeitet auch auf bescheidener Hardware zufriedenstellend und besticht durch die für Unix typische Netzwerkfähigkeit und Mehrnutzerunterstützung. Dies sind Funktionen, die am Desktop – vor allem am häuslichen – nicht oder nur selten benötigt werden. Linux ist deshalb bislang und wohl auch in der Zukunft keine Alternative am Arbeitsplatz. Die oft bemängelte Unterstützung seltener Hardware, wie Soundkarten oder Video, ist hinderlich für einen erfolgreichen Einbruch in die Microsoft-Domäne Desktop.

Eine neue Welt

Was spricht für Linux?

Ein Argument ist der Preis für die Software-Lizenz. Außer der für etwa 100 DM erhältlichen Distribution fällt bei Freeware nichts weiter an. Werden davon einige Dutzend Server installiert, ist der Unterschied zu jedem kommerziell verfügbaren System erheblich. Der erklärte Feind der Linux-Jünger ist Windows. Linux verfügt über Stärken im Netzwerkbereich und die sind beim Einsatz als Webserver besonders gefragt. Die aktuellen Statistiken für den Einsatz von Linux (siehe Abbildung 1.2) zeigen dies deutlich. Erstaunlich ist dabei, dass die Entwicklung sich offenbar nicht zu Ungunsten der bestehenden Windows NT-Server vollzieht, sondern den kommerziellen und teuren Unixen, allen voran Marktführer Sun (Solaris), das Wasser abgräbt (siehe Tabelle 1.1). Daran ändert auch die emotional hochgepeitschte Stimmung gegen Microsoft nichts. Jeder kommerzielle Anbieter eines Unix will seinen Markt beherrschen und Geld mit Services verdienen, die erst durch sein System notwendig werden. Genau an dieser Stelle sollte man auch Linux kritisch betrachten.

Abbildung 1.2:
Verteilung der
Betriebssysteme für
.de-Domains



Quelle: The Internet Operating System Counter, www.leb.net/hzo/ioscount/; (4/ 1999)

Betriebssystem	01/99	04/99	Veränderung
Linux	28,5 %	31,3 %	+ 2,8 %
Windows NT	24,4 %	24,3 %	- 0,1 %
Solaris	17,5 %	16,7 %	- 1,0 %
BSD	15,0 %	14,6 %	- 0,4 %
IRIX	5,3 %	4,6 %	- 0,7 %
Apple	1,6 %	2,1 %	+ 0,5 %

Tabelle 1.1:
Veränderung der
Betriebssystem-
nutzung für
Webserver zwischen
Januar 1999 und
April 1999 (globale
Nutzung mit .edu-
Domains)

Quelle: The Internet Operating System Counter www.leb.net/hzo/ioscount/; (4/1999)

Kostenlos ist nicht umsonst

Linux-Server kosten Geld, und das nicht zu knapp. Der Kaufpreis ist eine einmalige Angelegenheit und das Betriebssystem ist nur ein Teil der Investition. Auch wenn viele Programme für Linux ebenfalls unter der GPL veröffentlicht werden – sie installieren und pflegen sich nicht von selbst. Unix ist ein anspruchsvolles System, von und für Softwareentwickler und Administratoren entwickelt. An dieser Ausrichtung ändern auch grafische Aufsätze wie beispielsweise KDE nichts.

Die Notwendigkeit umfangreichen Fachwissens ist denn auch das eigentliche Problem im Umgang mit Linux. Spezialisten kosten bekanntlich Geld. Und wenn jemand meint, er sei ein Spezialist und arbeite umsonst, dann ist die Leistung meist auch entsprechend. Kritische Applikationen sind aber kein Spielgegenstand für in der Ausbildung befindliche Administratoren. Ein Linux-Server kann im Laufe der Zeit teuer werden. Vor allem beim ständigen Zugriff auf entsprechende Fachleute. Dies berücksichtigen die Freaks natürlich nicht, denn sie beherrschen das System und halten endlose Kommandozeilen oder schnell entworfene Batch-Skripte für normal.

Erinnern Sie sich an die Diskussionen über DOS und den Ärger über die »DOSen«-Computer? Wegen der ungeliebten Kommandozeile und der kryptischen Bedienung belächelten stolze Besitzer eines Apple-Computers die Microsoft-Anhänger. Das Lächeln wurde schmaler, als Windows seinen Siegeszug antrat. Es dauerte jedoch nur ein paar Tage, da war klar: Windows ist nur ein Aufsatz für DOS, kein richtiges Betriebssystem, eben nur Show. Die Diskussion mag heute niemand mehr wiederholen, dennoch erinnert man sich bei Linux manchmal daran. Das eigentliche Betriebssystem kennt nur den Prompt – weißer blinkender Cursor auf schwarzem Grund. Alles andere sind mehr oder weniger gelungene und mehr oder weniger zueinander kompatible Aufsätze – iMac-Besitzer müssten vor Lachen keine Luft mehr bekommen.

Was kostet Linux?

Es sind jedoch handfeste technische Parameter, die den Einsatz von Linux als Server zu einer sinnvollen Investition werden lassen.

Auf dem Weg zum Markt

Das Internet steht mit allen seinen Ausprägungen und Chancen noch ganz am Anfang. Linux ist ein typischer Wegbereiter für einen jungen Markt. Es ist leicht zu testen, steht in vielen Distributionen zur Verfügung und wird oft mit einer ganzen Palette ebenso preiswerter oder kostenloser wie wertvoller Software gebündelt. Linux wird von vielen Entwicklern programmiert, einzig der Kernel – die »Kernsoftware« – wird von den ursprünglichen Entwicklern der ersten Version, allen voran Erfinder Linus Torvalds, gezielt freigegeben.

Für die jungen Start-Ups, allen voran die vielen Internet Service Provider (ISP), ist Geld ein entscheidender Faktor. Für einen so jungen Markt ist der enorme Preisverfall der Dienstleistungen eine große Bedrohung, jede Mark ist entscheidend. Hier hat Linux und die reichhaltige Open Source-Software, wie der Apache-Webserver, einen Vorteil. Dies führte zu der enormen Verbreitung im Bereich der Webserver. Und dieselben Firmen werden oft auch von den Spezialisten gegründet, die sich aufgrund ihres überragenden Fachwissens für fähig genug halten, ein solches System zu beherrschen. Sie kennen sich auch mit dem Medium Internet perfekt aus, können die vielen Quellen für Hilfe und Unterstützung finden und nutzen.

Eben diese Unterstützung macht letztendlich Microsoft, Sun und HP mit ihren Systemen so stark, egal ob diese auf NT oder Unix basieren. Es ist nicht der oft zu Recht gescholtene Microsoft-Support, der hier hervorgehoben werden soll. Es ist die Ausbildungslinie, die ein Produkt so stark macht. Wenn Sie einen Experten für Windows NT brauchen, achten Sie auf sein MCSE-Zertifikat. Ähnliches vergibt auch Sun, bekannt für sehr teure, aber auch ausgesprochen professionelle Kurse. Es ist gut zu erkennen, wer Experte ist und wer sich nur als solcher bezeichnet. Man sollte Linux-Experten keineswegs unterstellen, sie wären mangels »Herstellerzertifikat« schlechter. Aber der unbedarfte Anwender, der sein mit viel Aufwand entwickeltes E-Commerce-System lauffähig halten muss, muss sich praktisch an Bestätigungen unbekannter Firmen orientieren. Trotzdem sind Linux-Experten inzwischen zu bekommen und der Support der Distributoren und angeschlossener Händler ist durchaus professionell. In diesem Umfeld kann Linux ebenso zum Einsatz gelangen wie jedes andere System. Sie sollten also wegen der bereits erwähnten technischen Vorteile Linux unbedingt ernsthaft in Erwägung ziehen. Der geringfügig höhere Einarbeitungsaufwand lohnt sich allemal, solange der Schwerpunkt nicht der Desktop ist.

Der Autor selbst entwickelt unter Windows und nutzt als »Produktionsserver« Linux – eine ebenso sinnvolle wie leistungsstarke Kombination.

Vor dem Einsatz

Vor dem Aufbau eines E-Commerce-Systems steht nicht nur die Wahl der Software, sondern die Wahl des gesamten Systems. Der Einstiegslevel bringt dabei wenig Auswahlmöglichkeiten. Mietsysteme laufen auf irgendeiner Plattform und werden per Browser bedient. Im nächsten Schritt sind oft eigene Server erforderlich, die beim Provider stehen. Spätestens hier müssen Sie sich nun doch entscheiden. Die in letzter Zeit mit wechselndem Ergebnis grassierenden Vergleiche Windows NT versus Linux haben vor allem gezeigt, dass es keinen Sieger gibt. Die Frage des richtigen Betriebssystems ist also nicht eine Frage der Leistung auf einer bestimmten Hardware. Sicher gibt es Dutzende konstruierte Grenzfälle, wo das eine oder andere System klar vorne liegt. Für die Masse der Anwendungen spielt das aber kaum eine Rolle. Bleibt die Unterstützung durch eigene Administratoren und die Verfügbarkeit entsprechender Software.

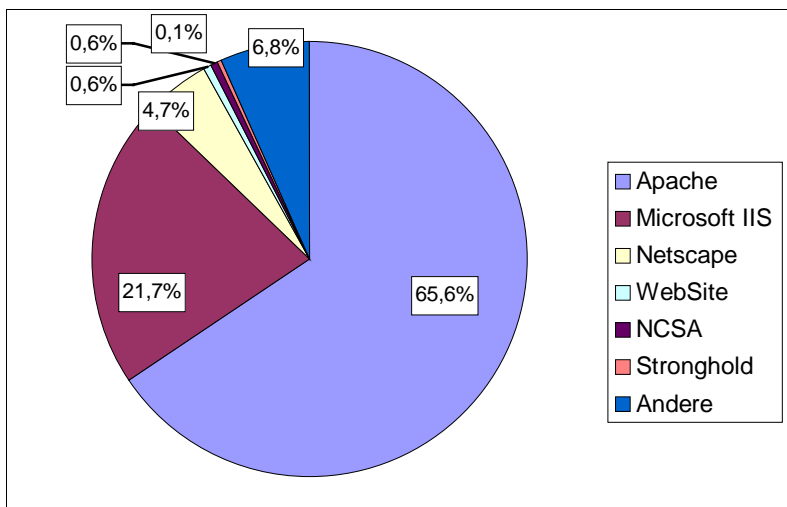


Abbildung 1.3:
Anteile des Apache-
Webservers weltweit
und bei .de-Domains
(berücksichtigt nicht,
auf welchem
Betriebssystem)

Quelle: E-Soft Inc. www.e-softinc.com/survey

Viel kritischer ist die Installation eines kompletten Servers zu sehen. Während man sich bei NT in Richtung kommerzielle Software flüchten kann – für Geld gibt es praktisch alles – ist bei Linux der Griff zum C-Compiler angesagt. Der weltweit bei Webservern führende Apache (siehe Abbildung 1.3) ist durchaus allen Ansprüchen gewachsen, wie auch Linux auf fast allen Plattformen verfügbar ist. Statt teurer Module kann man bei Problemen den Quellcode modifizieren und neu übersetzen. Können Sie oder Ihre Experten das? Wenn ja, sollten Sie

Linux ernsthaft in Betracht ziehen. Sie werden ein schnelles, stabiles, modernes und leistungsfähiges Betriebssystem für Ihr E-Commerce-Engagement nutzen können. Wenn nicht, sollten Sie sich die 99 DM für eine der vielen Distributionen und viele Tage verzweifeltes Probieren und Testen sparen.

Die Frage, ob Linux wirklich eine Alternative ist, kann man nicht direkt beantworten. Radio Eriwan würde antworten: »Im Prinzip schon, aber ...«. Im Detail betrachtet, gibt es viele Alternativen und Varianten, die teils mit, teils ohne Linux vorkommen und alle ihre Berechtigung haben. Betrachten Sie Tabelle 1.2 als Checkliste für eine Entscheidungsgrundlage. Vielleicht finden Sie dann die Antwort, die das Gewissen beruhigt.

Tabelle 1.2:
Pro und kontra
Linux

Dafür	Dagegen
<ul style="list-style-type: none"> • kostenlos; gilt auch für Zusatzsoftware • Quellcode verfügbar • sehr gute Netzwerk- und Internetunterstützung • stabil und universell • extrem anpassungsfähig • viele Dienstprogramme 	<ul style="list-style-type: none"> • komplizierte Installation (wenn vom Standard abweichend) • grafische Oberfläche schwach • kryptische Bedienung (Shell) • nicht mit Windows-Software kompatibel • Support nur via Internet, meist nur in Englisch verfügbar

1.7 Was ist Open Source?

An einigen Stellen war schon von Open Source, GPL und anderen Begriffen die Rede, die aus der Welt der freien Software stammen. Wenn Sie bislang Software teuer bezahlt haben, werden Sie sich wundern, wie so etwas überhaupt funktioniert. Wieso gibt es offensichtlich sehr fähige Leute, die Software entwickeln und dann verschenken?

Linux, PHP, der Apache-Webserver und nicht zuletzt MySQL sind kostenfrei erhältlich, wenn man von den Kosten für das Herunterladen oder die CD außer Acht lässt. Aber nicht nur das, auch die Quelltexte sind frei verfügbar. Der Quellcode von Windows ist das größte und bestgehütetste Geheimnis vom Microsoft. Warum also geben andere dieses umfangreiche Know-how einfach so preis?

Open Source-Software wird nicht von einer zentralen, kommerziellen und gewinnorientierten Firma hergestellt, sondern von vielen freiwilligen und enthusiastischen Programmierern. Die Lizenzen, die der Software beiliegen, erlauben nicht nur die Verteilung und Verbreitung, sondern auch die Modifikation des Codes, die Anpassung, Verfremdung und Einbau in eigene Lösungen. Einzig der Weiterverkauf ohne Bezug auf die ursprüngliche Lizenz ist nicht erlaubt.

Dabei spricht auch aus kaufmännischer Sicht einiges für freie Software. Statt für teure Lizenzen müssten sich Softwarefirmen nur für die tatsächlich erbrachten Leistungen bezahlen lassen. Es stünden dem Kunden bei gleichem Budget mehr Stunden an Programmierarbeit zur Verfügung. Wenn Sie mit Hilfe von PHP einen Shop programmieren, werden Sie diese Eigenleistung natürlich nicht verschenken, sondern sich angemessen bezahlen lassen. Müssen Sie erst selbst Tausende Mark für Lizenzen bezahlen, wird Ihre Leistung vielleicht für den Kunden zu teuer. Open Source kann also Ihr Geschäft durchaus fördern. Im Linux-Bereich haben sich bereits viele Dienstleister gefunden, die fehlenden Support einer freien Software ersetzen. Die Distributoren bieten ihre CDs und Handbücher zusammen mit Installationssupport und weitergehenden Leistungen an. Sie bezahlen dabei nicht für Linux, sondern für diese zusätzlichen Leistungen. So ist trotz des effektiv sehr geringen Preises eine gute Unterstützung möglich. Kommerzielle Softwarefirmen bieten oft weniger Hilfe für deutlich mehr Geld an.

Wenn Sie selbst Software entwickeln, die Sie auf diesem Wege vertreiben möchten, schauen Sie sich die übersetzte GPL-Lizenz an, die Sie im Buch am Anfang des Referenzteils finden. Sie können dann eher die Idee verstehen, die dahintersteckt. Beispielsweise wurde die beiliegende Referenz selbst unter der GPL veröffentlicht und nicht unter dem Copyright des Verlags. Im Übrigen ist auch die Software in diesem Buch, soweit sie vom Autor stammt, frei verwendbar, sowohl im Quelltext als auch in der fertigen Applikation. Beachten Sie aber, dass viele Codes aus dem Internet stammen und auf der CD immer die beiliegenden Lizenzen der ursprünglichen Autoren gelten.

1.8 Tipps zum sauberen Programmieren

Sicher hat jeder Programmierer seinen Stil und kennt viele mögliche Varianten für eine saubere Programmierung. Anfänger sollten sich jedoch nicht von fremden stilistischen Formen leiten lassen, ohne den Sinn zu erkennen, der hinter der einen oder anderen Schreibweise steckt. Die besonderen Anforderungen des Webs sind auch nicht jedem Profi völlig vertraut. Die folgenden Tipps zeigen, worauf es ankommt. Eine auch optisch ansprechende Codierung wird erreicht, wenn Sie:

- Code-Konventionen einhalten und
- Formatierung und Strukturierung beachten.

Mehr dazu finden Sie in ➡ Abschnitt 3.5.4 *Benutzerdefiniertes Fehlermanagement* ab Seite 274.

**Werden Sie ein
»guter«
Programmierer!**

Trennen Sie Code vom Design

Die Trennung hilft, leicht beherrschbare Programmteile zu erhalten. Die Wiederverwendbarkeit des Codes wird gesteigert. Sie können mit Gestaltungswerkzeugen arbeiten, die eingeschlossenen Code nicht verstehen. In großen Teams können Sie das Design von Designern erledigen lassen und die reine Programmierarbeit Programmierern übergeben.

Zur Wiederverwendbarkeit von Design und Code

Wenn Sie etwas ändern möchten, müssen Sie es nur an einer Stelle tun. Konsistente Gestaltungsmerkmale und Bedienerführung erleichtern den Nutzern die Navigation. Behandeln Sie Design-Elemente als Komponenten, die Sie immer wieder verwenden.

Verwenden Sie keine komplexen URLs

URLs mit vielen Parametern zeigen Hackern die innere Struktur Ihrer Applikation und legen Angriffspunkte offen. Nutzer werden vielleicht auch Ihre Seite als Lesezeichen ablegen. Wenn Sie die innere Struktur ändern, verlieren Sie Leser, die über die Lesezeichenverwaltung des Browsers auf Ihre Site zugreifen.

Personalisieren Sie Ihre Seiten

Sie steigern damit die Verbindung zum Nutzer und animieren ihn, wieder zu kommen. Wer Fragen doppelt stellt, gilt als dumm.

Unterstützen Sie Proxies und Caches

Nicht jeder Nutzer hat T-DSL oder 2-Mbit-Festverbindungen. Wenn der Browser-Cache unterstützt wird, verkürzen Sie aktiv die Ladezeiten. Dynamisieren Sie nur die Seiten, bei denen es wirklich sinnvoll ist.



Vorbereitung und Installation

In diesem Kapitel lesen Sie alles über die Installation unter Linux und Windows und die Vorbereitung einer Entwicklungsumgebung. Vermittelt werden außerdem Informationen zur Nutzung von PHP in gemietetem Webspace. Wer neu in der Webserverprogrammierung ist, sollte außerdem die Grundlagen in den ersten beiden Abschnitten lesen, um mehr über die technischen Hintergründe zu erfahren.

Netzwerkgrundlagen (Seite 47)

Höhere Netzwerkprotokolle (Seite 72)

Vorbereitung der Installation (Seite 106)

Installation unter Windows NT/2000 (Seite 107)

Installation unter Linux (Seite 114)

PHP konfigurieren (Seite 119)

Sicherheit (Seite 135)

PHP im Webspace (Seite 137)

2.1 Netzwerkgrundlagen

Um den Sinn der dynamischen Seiten zu verstehen, ist für alle Einsteiger ein kurzer historischer Rückblick zu empfehlen. Um die Beispiele zu verstehen, sollten Sie über elementare Kenntnisse in HTML verfügen.



2.1.1 Rückblick

Am Anfang bestand das World Wide Web (WWW), auch aufgrund seiner ursprünglichen Definition, aus statischen, miteinander verlinkten Seiten. Auch heute noch sind die allermeisten Präsenzen so aufgebaut. Als Protokoll zwischen Webserver und Browser dient das *Hypertext Transfer Protocol* (HTTP). Über dieses Protokoll fordert der Browser eine bestimmte Seite an, und der Server reagiert darauf mit der Übertragung der entsprechenden Seite oder einem Fehler, wenn die Seite nicht gefunden wurde. Hyperlinks sind Verweise in statischen Webseiten, die auf andere Dateien im Web zeigen und solche Übertragungsprozeduren auslösen. Mit der Hypertexttechnik kann man durchaus anspruchsvolle Webseiten erstellen. Der Schwerpunkt der Entwicklung lag jedoch auf der puren Informationsvermittlung. Hyperlinks erlauben es Dokumenten, den Leser in gewisser Weise zu führen. Der Zugriff auf verwandte Informationen erleichtert die Verarbeitung von Inhalten. Hypertextsysteme sind für das Lesen von Texten am Bildschirm gedacht und strukturieren Dokumente anders als Bücher. Von der grundsätzlichen Idee her ist damit aber die Zielrichtung vorgegeben: Informationsvermittlung. Ein bekanntes Hypertextsystem ist beispielsweise auch die Windows-Hilfe. Mit der Entwicklung des Webs zu einem Medium der Werbung und Präsentation auf der einen Seite, zu einer direkten interaktiven Einbeziehung des potenziellen Kunden auf der anderen Seite war jedoch der klassische Hypertext, geschrieben in der *Hypertext Markup Language* (HTML), den Anforderungen nicht mehr gewachsen. Aufgrund der Historie sind nie ernsthafte Versuche unternommen worden, neben HTML einen neuen Standard für Webseiten zu kreieren, der die Hypertextstruktur um echte Interaktion ergänzt. So sind im Laufe der letzten Jahre die verschiedensten Systeme entstanden, die außerhalb HTML arbeiten und den Webservern die so dringend benötigte Interaktivität verleihen. Die bekanntesten Beispiele sind CGI-Programme und webservernspezifische Applikationen, die spezielle Schnittstellen benutzen. Das *Common Gateway Interface* (CGI) stellt eine genormte Schnittstelle zwischen den CGI-Programmen und dem Webserver dar. Häufig wird dazu die Skriptsprache Perl eingesetzt. Perl ist als Interpreter konzipiert (wie alle Skriptsprachen) und die Skripte laufen als eigenständige Programme ab, gestartet durch eine Anforderung des Browsers. Das Ergebnis können Zugriffe

Was ist Hypertext?

Das Ergebnis können Zugriffe auf interne Funktionen des Servers oder Ausgaben in HTML sein. Die Programmierung ist verhältnismäßig komplex und die Erzeugung graphisch ansprechend gestalteter HTML-Seiten extrem aufwändig, denn jeder einzelne HTML-Befehl wird einzeln erzeugt und dann zum Webserver gesendet. Auch die Erzeugung von speziellen, nativen Programmen, die auf die Interfaces der Webserver direkt zugreifen, ist kompliziert und der professionellen Programmierung vorbehalten. So stellt Microsoft mit dem Internet Information Server das Information Server Application Programming Interface (ISAPI) zur Verfügung, Netscape mit seinen Webservern das Gegenstück Netscape Server Application Programming Interface (NSAPI). Praktisch sind APIs Sammlungen von Bibliotheksfunktionen (Dynamic Link Libraries, DLL), auf die Programmierer zurückgreifen können, die Anwendungen in richtigen Programmiersprachen schreiben, beispielsweise C++. Auch diese Programme erzeugen HTML aber nur seriell über eine bestimmte Ausgabeprozedur, wenngleich sich damit natürlich schnelle und komplexe Anwendungen entwickeln lassen.

2.1.2 Ausblick

Für den Praktiker, der als Webdesigner oder Webprogrammierer schnell umfangreiche Anwendungen für das Internet produzieren muss, sind CGI-Applikationen oft zu aufwändig. Auch sind bei dem heute stark präsentationsgeprägten Web eher gute Multimediadesigner gefragt als der klassische Programmierertyp. Trotzdem sollen die Webseiten interaktiv sein, auf Aktionen des Nutzers also reagieren können, es sollen Zugriffe auf Datenbanken möglich sein, und die Eingaben der Nutzer sollten ebenso in Datenbanken gespeichert werden können. Eine solche einfache, aber zugleich leistungsfähige Schnittstelle stellen die Skriptsprachen dar, die direkt in die HTML-Seiten eingebunden werden. Eine der effizientesten ist PHP.

Um mit PHP arbeiten zu können, sind im Vorfeld also nur HTML-Kenntnisse nötig, und das auch nur in dem Umfang, wie es die gewünschten Webseiten erfordern. Wer anspruchsvolle Seiten mit Cascading Style Sheets, Dynamic-HTML und JavaScript erstellen will, sollte diese Elemente beherrschen. PHP bietet dazu nur selten eine Alternative. PHP ist kein Gestaltungselement. Was der Browser sieht, ist pures HTML mit seinen diversen Erweiterungen, Verbesserungen und irgendwann sicher Nachfolger wie XHTML. Aber PHP ist in HTML integriert und ordnet sich der Logik der HTML-Seite unter. Und PHP benötigt keine bestimmte browserseitige Technik, kann also mit jedem Standardbrowser arbeiten.



Tip

Mit der Kombination aus MySQL und PHP haben Sie die Technologie, um datenbankgestützte Websites zu erstellen.

2.1.3 Grundbegriffe

Lernen Sie die Grundbegriffe kennen, die im Zusammenhang mit dem Internet immer wieder genannt werden. Das Verständnis der Technik hilft, Probleme bei der Programmierung zu erkennen und die Reaktionen der Server zu verstehen. Vorgestellt werden auch die wichtigsten Protokolle auf einfachem Niveau. Bei der Erläuterung der darauf direkt Bezug nehmenden Funktionen werden diese Grundlagen weiter vertieft, insofern ist ein wenig Redundanz beabsichtigt.

Aufbau des Internet

Das Internet als Begriff ist keine fassbare Einheit, die in einem Stück zu beschreiben wäre. Ich will an dieser Stelle eine kurze Erklärung versuchen, die das Internet und seine grundsätzliche Funktionalität verständlich machen.

Das Internet gehört niemandem und wird von niemandem kontrolliert. Als Internet wird die Einheit aller Personen, Firmen, Organisationen verstanden, die sich unter Einhaltung bestimmter Standards und Normen zusammenschließen. Das können auf der einen Seite Informationsanbieter (die Server) sein, auf der anderen Seite gehören auch alle Informationsnutzer (die Surfer) dazu. Alle sind durch Datenleitungen miteinander verbunden. Für den privaten und kleineren gewerblichen Teilnehmer bestehen diese Datenleitungen aus normalen Telefonleitungen, die von Telekommunikationsfirmen gemietet werden. Die Struktur des Internets bilden sogenannte Provider (Verteiler), die Knoten bereitstellen, an denen die Datenströme gesammelt und neu verteilt werden. Physisch sind diese Knoten die Punkte, an denen sich die Nutzer mit Modem oder ISDN-Karte einwählen und dann von den Computern (Gateways, Router) des Providers mit den anderen Leitungen des Internets verbunden werden. Provider, die solche Einwählpunkte betreiben, werden auch PoPs³ genannt. Manche Provider haben viele selbstständige PoPs, betreiben selbst aber nur die Hauptleitungen (Teile des Backbone). Man unterscheidet echte Serviceprovider, die solche Leitungen nur mieten und darauf ausschließlich Datendienste betreiben, und sogenannte Carrier, die selbst Überlandleitungen verlegen und für alle Dienste – Sprache und Daten – vermieten. Zwischen allen Städten der Welt existieren Dutzende von privaten oder staatlich kontrollierten Leitungen. Damit können die Daten auch auf verschiedenen Wegen durch die Welt gelangen.

Das Internet ist deshalb nicht unbedingt schnell, aber außerordentlich sicher. Auch ein totaler Schnitt durch alle Leitungen einer Zone würde

³ Vom englischen Point of Presence; etwa: Punkt, an dem das Internet präsent ist.

den Informationsfluss nicht zum Stillstand bringen; die Daten gehen dann eben längere Wege, aber sie erreichen sicher ihr Ziel.

Jeder einzelne Computernutzer kann Bestandteil des Internets werden, indem er einen Server betreibt, den alle anderen Teilnehmer sehen können. Die einzige gemeinsame Basis ist die Verwendung bestimmter Standards. Für das Netzwerk ist dies TCP/IP als Protokoll und für die Informationsinhalte sind es HTTP (WorldWideWeb), FTP (Dateitransfer), NNTP (News) und SMTP (E-Mail).

Jeder kann also behaupten: »Ich bin das Internet« (oder wenigstens ein kleiner Teil davon). Wenn Sie zur Zeit einen Server installieren, sind Sie ungefähr ein 10millionstel des Internets, denn so viele Server gibt es mittlerweile.

Sie müssen nicht einmal einen eigenen Server betreiben, denn Sie können auch Speicherplatz bei einem anderen Server für Ihre Seiten mieten. Das nennt man dann Webspace. Diese Lösung ist zwar nicht sonderlich flexibel, aber recht billig. Allerdings sind die in diesem Buch gezeigten Techniken mit starken Eingriffen in die Maschine verbunden. Deshalb empfehle ich dringend, einen eigenen Server anzuschaffen.

World Wide Web	Das WWW ist der Multimedia-Dienst des Internets. Im WWW werden HTML-Dokumente platziert, die aus Text, Bildern, Animationen, Video- und Soundsequenzen etc. bestehen können. Ein großer Vorteil des WWW besteht darin, dass auf andere, weiterführende Dokumente über sogenannte Hyperlinks (Verweise) bequem zugegriffen werden kann. Das WWW ist der Dienst des Internets, der eigentlich gemeint ist, wenn Sie in den Medien auf den Begriff Internet stoßen. Das WWW ist zwar nur ein Teilbereich des Internets, jedoch wohl derjenige mit der höchsten Expansionsrate.
HTML	HTML ist die Beschreibungssprache, mit der die Dokumente codiert werden. Der Name <u>H</u> ypertext <u>M</u> arkup <u>L</u> anguage deutet auf den ursprünglichen Zweck hin: die Einbettung von Verknüpfungen zu weiterführenden Informationen. So ist das WWW weltweit in vielfältiger Weise miteinander verknüpft und verflochten.
TELNET	<p>Telnet (<i>Virtual Terminal Protocol</i>, VTP) ist der Dinosaurier des Internets. Mit Telnet können Sie von Ihrem Computer über das Internet auf einen anderen Computer zugreifen.</p> <p>Sie können – wenn Sie über die entsprechenden Rechte verfügen – mittels Internet auf einem entfernten Rechner arbeiten und die entsprechenden Ressourcen (zum Beispiel Drucker) nutzen, als würden Sie direkt an jenem Rechner sitzen. Für den Einsatz im Windows NT-Server wird TELNET normalerweise nicht benötigt.</p>

Das File Transfer Protocol (FTP) ermöglicht Ihnen den Zugriff auf bestimmte Dateien innerhalb des Internets. Sie können mit dem FTP einerseits Daten von einem entfernten auf Ihren Computer herunterladen, andererseits von Ihrem Computer Daten auf einen anderen Computer übertragen. Das FTP ist eine Art Datei-Kopiermöglichkeit. FTP-Serveradressen beginnen mit »ftp:«.

FTP

Auf direktem Wege können Sie keine Daten per FTP in eine Datenbank laden. Aber mit Hilfe der in diesem Buch vorgestellten Techniken ist es möglich, große Datenmengen per FTP auf einen Webserver zu übertragen und dort in die Datenbank zu importieren.

Electronic-Mail (auch E-Mail⁴) ist die elektronische Post, die von einem Computer zu einem anderen, von einem Nutzer zu einem anderen verschickt wird. Wenn Sie einen Internetzugang besitzen, haben Sie in der Regel auch eine oder mehrere E-Mail-Adressen. E-Mail bietet im Gegensatz zur »klassischen Post« gleich mehrere Vorteile: E-Mail ist wesentlich schneller. In der Regel erhalten Sie eine E-Mail bereits wenige Augenblicke nachdem sie abgeschickt wurde (vorausgesetzt Sie schauen auch in Ihr »Postfach«), und zwar egal, von wo aus sie abgeschickt wurde (ob aus New York, Tokio, Berlin oder Buxtehude). E-Mail ist geographisch unabhängig. Nehmen wir an, Sie befinden sich auf Geschäftsreise. Ihre Sekretärin hat wichtige Post für Sie. Normalerweise würde Ihre Mitarbeiterin versuchen, die Telefaxnummer Ihres Hotels herauszubekommen und Ihnen die Post per Fax zukommen lassen oder Sie telefonisch informieren. Bei E-Mail ist es anders. Sie müssen einfach nur in Ihr »Postfach« schauen, ob eine neue Nachricht vorliegt.

E-Mail

E-Mail ist der bedeutendste Dienst im Internet. So wurden 1998 nach Schätzungen 3,4 Billionen E-Mails versendet.



Hinweis

Das Internet Relay Chat (IRC) ist wie E-Mail ein Online-Dienst. Im Gegensatz zur E-Mail, bei der Sie von einem Nutzer Post bekommen und ihm dann im Gegenzug eventuell Post zurücksenden, können Sie beim IRC »live« mit anderen Internetteilnehmern konferieren. Dabei ist es unerheblich, wo sich die Teilnehmer gerade aufhalten, ob sie sich im Nachbarzimmer oder aber tausende Kilometer weit entfernt befinden. Jede Eingabe Ihrer Tastatur wird sofort an Ihren Gesprächspartner übertragen und umgekehrt. So können Sie sich sofort und unmittelbar mit anderen Teilnehmern zu bestimmten Themen austauschen. Für IRC werden spezielle Programme auf dem Computer benötigt, die es im Internet kostenlos gibt.

Internet Relay Chat (IRC)

⁴ Laut Duden ist die Schreibweise »E-Mail« die einzige zulässige Variante. Alle anderen Schreibweisen, wie *eMail* oder auch *Email* sind unrichtig (Der Duden, Bd. 1, 21. Auflage 1996; Dudenverlag).

Newsgroups	Die Newsgroups sind eine Mischung zwischen E-Mail und IRC. Hier wird öffentlich, das heißt, mit mehreren Teilnehmern weltweit diskutiert. Die Newsgroups sind mit einem Schwarzen Brett vergleichbar. Jemand trägt zu einem bestimmten Thema eine »Wortspende« bei, die in der entsprechenden Newsgroup veröffentlicht wird. Alle anderen Teilnehmer können dann auf den entsprechenden Beitrag antworten bzw. eigene Beiträge leisten. Newsgroups gibt es zu den verschiedensten Themenbereichen. Vermutlich existieren weltweit einige hunderttausend Newsgroups. Große Newsserver liefern 30 000 bis 50 000.
Gopher	Gopher ist ebenfalls ein WWW-Dienst. Im Gegensatz zu den üblichen WWW-Seiten, die als HTML-Dokumente gestaltet werden, ist Gopher jedoch mehr eine Liste mit Aufzählungen von Menüpunkten, die dann auf entsprechende Themen wie bei HTML-Dokumenten mit Hyperlinks verweisen. Gopher hat durch das WWW stark an Bedeutung verloren, wenngleich auch heute noch viele Gopher-Server erreichbar sind. Windows NT-Server 4 unterstützt Gopher nicht mehr.
WAIS	Eine <i>Wide Area Information System</i> -Datenbank (WAIS-Datenbank) ist ein durchsuchbares Dokument. Hierfür existiert eine definierte Abfragesprache. Das Ergebnis bzw. die Trefferliste, die von einem WAIS-Server zurückgegeben wird, ist ein Hypertext-Dokument mit Links auf die gefundenen Dokumente.

Intranet und Extranet

Das große Schlagwort in der letzten Zeit ist das Intranet. Datenbanken bieten eine gute Möglichkeit, ein Intranet kräftig »aufzupeppen« oder überhaupt erst sinnvolle Funktionen einzubauen.

Intranet	Das Intranet ist ein firmeninternes Netzwerk, das Standards (Protokolle, Normen), Techniken (Gateways, Router), Infrastrukturen (Leitungen, Provider) und Angebote (fremde Server) des Internets nutzt, um firmeninterne Informationen zu verteilen. Der große Vorteil ist die Nutzung einer weltweiten, preiswerten und allgegenwärtigen Infrastruktur. Durch die Nutzung der Standards stehen mehr Fachleute und preiswertere Softwarelösungen zur Verfügung. Die Inhalte eines Intranets sind im Internet normalerweise nicht direkt sichtbar. Wenn Sie versuchen, einen Intranetserver einer Firma anzusprechen, dann werden Sie abgewiesen. Der Computer, der für die Abweisung zuständig ist, wird Firewall genannt.
Extranet	Ein Extranet ist ein geschlossenes Netz zwischen verschiedenen Firmen unter Nutzung des Internets. So könnten ein Hersteller und seine Vertragshändler in einem Extranet miteinander verbunden sein. Alle Teilnehmer an einem Extranet nutzen lokale Internetprovider für die nötigen Dienste.

Intranet und Extranet sind logische Einheiten und Komponenten, die das Internet nutzen.

Das Netzwerkprotokoll TCP/IP besitzt heute als das Standardprotokoll des Internet eine fundamentale Bedeutung. In diesem Kapitel sollen die Grundlagen besprochen werden, die für die richtige Planung und Umsetzung einer Windows 2000 Webserverlösung für den Einsatz im Internet oder im Intranet notwendig sind.

2.1.4 Die Internet-Protokolle und ihr Ursprung

Im Zusammenhang mit TCP/IP werden Sie häufig auf die Begriffe Internet Protocol Suite (IPS), Internet-Protokollfamilie oder einfach Internet-Protokolle stoßen. Dies sind eigentlich die heute üblichen Bezeichnungen. Im folgenden Text werden wir vor allem die deutschen Begriffe einsetzen.

Internet Protocol Suite (IPS)

Daneben werden die Internet-Protokolle auch als Department of Defense (DoD)- oder ARPANET-Protokolle bezeichnet. Dies ist in der Entstehungsgeschichte des Internet begründet, dessen Entwicklung Ende der sechziger Jahre durch das amerikanische Verteidigungsministerium initiiert wurde und im Aufbau des ARPANET (*Advanced Research Project Agency*) gipfelte. Das ARPANET stellt damit die erste Implementierung des Internet mit seinen wesentlichen Protokollbestandteilen dar. Noch heute ist es, natürlich in weiterentwickelter Form, eines der größten zusammenhängenden IP-Netzwerke.

DoD / ARPANET

Der Beginn der Entwicklung plattformübergreifender Kommunikationsmöglichkeiten zwischen Computersystemen liegt in einer Zeit, als an allgemein gültige Standards in diesem Bereich noch nicht zu denken war. Mangels Alternativen entschlossen sich aber nach öffentlicher Verfügbarkeit der Internet-Protokollfamilie immer mehr Hersteller, diesen Quasi-Standard zu unterstützen und kompatible Netzwerklösungen zu entwickeln.

2.1.5 Die Organisationen ICANN und IANA

Für die Festlegung und Sicherung eines für das Internet allgemeingültigen Domain-Standards zeichnen vor allem zwei Institutionen verantwortlich:

- ICANN (Internet Corporation for Assigned Names and Numbers) **ICANN**

Diese Organisation wurde erst im Oktober 1998 gegründet und dient der Sicherstellung der technischen Rahmenbedingungen für das Internet. Sie zeichnet für die grundlegenden Kommunikationsstandards verantwortlich, welche vormals allein durch die US-Regierung beziehungsweise durch von ihr beauftragte Organisati-

onen wahrgenommen wurden. So obliegt ihr die Koordination des gesamten Internet-Rootserversystem. Darüber hinaus werden innerhalb dieses Gemiums Vorschläge der Internet-Community aufgegriffen und diskutiert.

Die Verwaltung der generischen TLDs (gTLDs wie COM, NET etc. liegt ebenfalls in der Verantwortung der ICANN. So werden hier die Unternehmen registriert und zugelassen, die für die Vergabe von SLDs direkt unterhalb der TLDs weltweit verantwortlich sind.

Weitergehende Informationen finden Sie auch auf der ICANN-Website:

www.icann.org

IANA

- IANA (Internet Assigned Number Authority)

Die IANA ist wie die ICANN eine Non-Profit-Organisation und dient der Sicherstellung der korrekten Vergabe der IP-Nummern und der zentralen Verwaltung der Country Code Top-Level Domains (ccTLDs wie US, DE).

Weitere Informationen finden Sie auch auf der Website der IANA:

www.iana.org

2.1.6 Das ISO/OSI-Modell und die Internet-Protokolle

International Organization for Standardization

1977 begann die ISO (International Organization for Standardization) mit den Arbeiten an einem allgemeingültigen Standard für die Kommunikation mit (und zwischen) Computersystemen. Im Ergebnis dieser Bemühungen entstand das heute bekannte und für die Beschreibung technischer Kommunikationsprozesse oft herangezogene ISO/OSI-Referenzmodell (*Reference Model for Open Systems Interconnection of the International Organization for Standardization*).

Das ISO/OSI-Referenzmodell

Dieses Modell teilt Netzwerkverbindungen in sieben logische Schichten ein, die jeweils eigene Aufgaben übernehmen. Eine höhere Schicht baut dabei immer auf den Funktionen der tiefer liegenden auf. Die folgenden sieben Schichten werden dabei unterschieden:

Tabelle 2.1:
Die sieben Schichten
des ISO/OSI-Referenzmodells

Nr.	Schicht	Funktionen
7	Anwendung	Nutzerschnittstelle, Kommando-Auswahl
6	Darstellung	Kodierung, Dekodierung, Kompression
5	Sitzung	Steuerung der Kommunikation
4	Transport	Verbindungsaufbau, Datentransport

Nr.	Schicht	Funktionen
3	Vermittlung	Adressierung, Routing
2	Sicherung	Fragmentierung, Kontrolle, Prüfung
1	Bitübertragung	Physischer Datentransport

Nachfolgend finden Sie, beginnend bei der untersten Schicht, die einzelnen Funktionen etwas genauer:

- Schicht 1: *Bitübertragungsschicht* (physical layer). Hier wird die physikalische Übertragung (elektrisch sowie mechanisch) definiert: das Medium (Kabel, Funk, Infrarot), die gesendeten Signale usw. **Bitübertragung**
- Schicht 2: *Sicherungsschicht* (data link layer, auch Verbindungsschicht oder MAC-Layer genannt). Hier werden die Daten in einzelne Rahmen aufgeteilt und gesichert übertragen. **Sicherung**
- Schicht 3: *Netzwerkschicht* (network layer, auch Vermittlungsschicht). Zentrale Aufgabe ist die Bestimmung eines optimalen Weges durch ein Netzwerk. **Vermittlung**
- Schicht 4: *Transportschicht* (transport layer). Diese Schicht stellt einen gesicherten Kanal zwischen zwei Stationen her, sodass die Daten einfach seriell geschrieben bzw. gelesen werden können. **Transport**
- Schicht 5: *Sitzungsschicht* (session layer, auch Kommunikationssteuerungsschicht). Diese Schicht synchronisiert das Zusammenspiel mehrerer Stationen. Es wird beispielsweise festgelegt, wie eine Sitzung zeitlich ablaufen hat (Aufforderung zum Senden eines Kennwortes, Senden des Kennwortes, Bestätigung des Kennwortes usw.). **Sitzung**
- Schicht 6: *Darstellungsschicht* (presentation layer). Hier werden die Daten in ein einheitliches Format transformiert, zum Beispiel durch Alphabetumwandlungen oder Datenkompression. An dieser Stelle gehen oft die Umlaute verloren, wenn die Übertragung mit 7 Bit statt 8 Bit erfolgt. **Darstellung**
- Schicht 7: *Anwendungsschicht* (application layer). Diese Schicht beschreibt die Schnittstelle, über die Anwendungen auf Dienste eines anderen Systems zugreifen können. **Anwendung**

Jede Schicht kommuniziert mit der entsprechenden Schicht auf dem anderen System (*logischer Datenfluss*), indem sie Daten entweder an die darüber oder darunter liegende Schicht weiterleitet (*physikalischer Datenfluss*). Dabei verfügt jede Schicht über Schnittstellen, die folgende Abläufe ausführen können: **Kommunikationsprozesse**

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

- Austausch von Daten mit der darüber liegenden Schicht
- Austausch von Daten mit der darunter liegenden Schicht
- Entscheidung darüber, welche Daten an dieselbe Schicht im anderen System übermittelt werden

Wenn die Sitzung auf Schicht 5 ihre Daten an die Schicht 4 übergeben hat, wartet sie, bis die Antwort von Schicht 5 des anderen Systems zurückkommt. Wie diese Nachricht auf das andere System gelangt, ist Aufgabe von Schicht 4, die sich wiederum nur mit Schicht 3 in Verbindung setzt, usw. Der wirkliche Datenaustausch findet nur auf Schicht 1 statt.

Durch dieses Verfahren sind höhere Schichten völlig unabhängig von den physikalischen Gegebenheiten (Funknetz, ISDN, Glasfaser usw.). Andererseits können über eine funktionierende physikalische Verbindung (Schicht 1) alle Arten von Daten und Protokollen (höhere Schichten) benutzt werden.

Abbildung der Internet-Protokolle im ISO/OSI-Referenzmodell

Der theoretische Ansatz des Referenzmodells geht davon aus, dass auf jeder Ebene ein Protokoll arbeitet. Allerdings trifft das gerade auf die Internetprotokolle nicht zu. Deren Entwicklung beginnt bereits, bevor die ISO am Referenzmodell arbeitet und verläuft praktisch parallel zu diesem.

4-Schichtenmodell der Internet-Protokollfamilie

Die Internet-Protokollfamilie kann aber durchaus mit dem ISO/OSI-Referenzmodell verglichen werden. Die Daten durchlaufen beim Transport über ein Übertragungsmedium, wie beispielsweise ein Kupferkabel, üblicherweise alle Schichten von der Anwendung des Senders bis hin zum Empfänger.

Kapselung der Daten

So übergibt eine Anwendung wie beispielsweise ein FTP-Client oder ein Terminalprogramm für Telnet seine Datenpakete an die Transportschicht. Hier bekommt das Paket einen Header, in dem weitere Informationen zu dessen Aufbau hinterlegt werden. Wird das Protokoll TCP verwendet, befinden sich im so genannten TCP-Header Angaben zum Quell- und Zielport sowie die TCP-Flags. Bei der Übergabe an die nächste Schicht (Netzwerk) wird das Paket um einen weiteren Header, beispielsweise den IP-Header, erweitert. In diesem werden unter anderem die IP-Quell- und Zieladresse hinterlegt, um den richtigen Weg im Netzwerk, auch über IP-Router, finden zu können. Schließlich erfolgt eine letzte Erweiterung des Pakets in der Verbindungsschicht. Der neue Header enthält dann unter anderem Informationen zum verwendeten Übertragungsverfahren wie Ethernet oder Token Ring. Beim Weg zum Empfänger werden dann alle Schichten rückwärts wieder durchlaufen und die jeweiligen Header-

Informationen entfernt. Dieser ganze Vorgang wird auch als Daten-Kapselung bezeichnet.

Mit Hilfe dieser Datenkapselung können Kommunikationslösungen geschaffen werden, welche unabhängig vom verwendeten technischen Verfahren funktionieren. So ist beispielsweise die Verwendung der IP-Protokollfamilie nicht an ein bestimmtes Übertragungsverfahren gebunden, sondern ist auch über Ethernet, Token Ring, ATM, PPP für die Datenfernübertragung oder andere, vielleicht erst in Zukunft verfügbare Medien möglich.

Die wichtigsten Protokollbestandteile der Internet-Protokollfamilie werden eingehender in folgenden Abschnitt beschrieben. Diese Protokolle werden auch bei der Programmierung von PHP von Bedeutung sein.

2.1.7 Request For Comments (RFC)

Als wichtiges Standardisierungsinstrument in der Protokollwelt sind seit mehr als 30 Jahren die so genannten RFCs in Gebrauch. RFCs (*Request For Comments*) sind öffentlich zugängliche Dokumente, die einem einheitlichen Schema und einer fortlaufenden Nummerierung folgen. In diesen können sich entsprechend qualifizierte Personen oder Hersteller äußern.

RFCs tragen generell eine fortlaufende Nummer. RFC 0001 wurde am 7. April 1969 veröffentlicht. Ändert sich ein RFC, wird eine neue Nummer vergeben und das alte Dokument als obsolet gekennzeichnet. Im vorliegenden Buch wird an einigen Stellen auf RFCs verwiesen. Damit können Sie, wenn Sie weitergehende Informationen benötigen, schnell die entsprechenden Quellen des betreffenden Standards oder der Technologie finden.

Stufen eines RFC

Ein RFC kann mehrere Stufen durchlaufen, bis es vielleicht einmal den »offiziellen« Status in Form eines anerkannten Standards erlangt. Eine der bekanntesten Standardisierungsgremien ist die IETF (*Internet Engineering Task Force*).

Mögliche Stufen eines RFC sind:

- Experimental (Experimentell)

Das hier spezifizierte Protokoll oder Verfahren sollte nur zu experimentellen Zwecken oder zur Evaluierung eingesetzt werden. Es sind noch grundlegende Änderungen möglich, ebenso wie das völlige Verwerfen der Entwicklung durch den Hersteller.

Experimental

Proposed Standard	<ul style="list-style-type: none">• Proposed Standard (Vorgeschlagener Standard) <p>Als Vorschlag werden RFCs gekennzeichnet, wenn die Standardisierung gezielt angestrebt wird. Dennoch befindet sich das Protokoll noch in der Entwicklung und wird voraussichtlich noch Änderungen unterworfen sein. Oft sind solche Änderungen auch Kompromisse, die notwendig sind, um die Anerkennung als Standard zu erlangen.</p>
Draft	<ul style="list-style-type: none">• Draft (Entwurf) <p>In diesem Stadium, das Sie häufiger beobachten können, befinden sich Dokumente, die als Standard ernsthaft in Betracht gezogen werden. Praktisch ist die Entwicklung abgeschlossen. Durch die Veröffentlichung gelangen die Methoden zum praktischen Einsatz. Im Feldtest können sich Probleme herausstellen, die noch zu Änderungen am endgültigen Standard führen.</p>
Standard	<ul style="list-style-type: none">• Standard <p>In dieser Phase ist das RFC verabschiedet und endgültig. Wenn sich Änderungen oder Weiterentwicklungen ergeben, wird eine neue Nummer vergeben und das alte RFC wird obsolet. Als Verabschiedungsgremium agiert das IAB (<i>Internet Architecture Board</i>).</p> <p>Neben diesen grundlegenden Eigenschaften können ergänzende Hinweise anfallen, die sich teilweise auch auf Systeme beziehen:</p>
Recommended	<ul style="list-style-type: none">• Recommended (empfohlen) <p>Das Protokoll wird zum Einsatz empfohlen.</p>
Not recommended	<ul style="list-style-type: none">• Not recommended (nicht empfohlen) <p>Es ist nicht empfehlenswert, dieses Protokoll einzusetzen, weil es oft inzwischen ein neueres gibt.</p>
Limited use	<ul style="list-style-type: none">• Limited use (begrenzter Einsatz) <p>Dieses Protokoll wird nur für sehr eng gesteckte Spezialfälle zur Anwendung kommen.</p>
Required	<ul style="list-style-type: none">• Required (erforderlich) <p>Zwingende Anwendung im Zusammenhang mit anderen Protokollen.</p>
Elective	<ul style="list-style-type: none">• Elective (wahlweise) <p>Für den vorgesehenen Zweck stehen mehrere Protokolle gleichwertig zur Auswahl.</p>
STD-Nummern	<p>Aus den gültigen, verabschiedeten RFCs werden Standards, indem eine Standardnummer STD zugewiesen wird. Manchmal umfasst ein</p>

solcher Standard mehrere RFCs. STD-Nummern sind endgültig, werden also nicht geändert, wenn sich die zugrunde liegenden RFCs verändern. Die Zusammenfassung der STDs und RFCs wird in der RFC 2500 spezifiziert.

Mehr Information und vor allem alle RFCs und STDs finden Sie im Internet unter den folgenden Adressen:

www.faqs.org
www.rfc-editor.org

RFCs im Internet

Im Buch wird an vielen Stellen auf die zum Thema passende RFC hingewiesen.

2.1.8 Internetprotokolle im Detail

In diesem Abschnitt werden die Protokolle der Internet-Protokollfamilie näher betrachtet. Dabei stehen vor allem die Protokolle im Mittelpunkt, die aus Sicht der Webserververwaltung wichtig sind.

Protokoll	Funktion	Seite
IP	Adressierung und Transport der Datenpakete (keine Fehlerkorrektur)	59
TCP	Gesicherter Transport der Daten mit Fehlerkorrektur	66
UDP	Ungesicherter Transport von Datenströmen ohne Fehlerkorrektur	68

*Tabelle 2.2:
Übersicht über die
behandelten Internet-
Protokolle*

Andere wichtige Protokolle der Schicht 4 (Anwendung) der Internet-Protokollfamilie wie SMTP oder HTTP werden eingehend in Abschnitt 2.2 *Höhere Netzwerkprotokolle* ab Seite 72 behandelt.

Internet Protocol (IP)

Das meistverwendete Protokoll auf der Schicht 2 (Netzwerk) der Internet-Protokollfamilie ist IP. Das wesentliche Merkmal dieses Protokolls besteht darin, dass jeder Netzwerkknoten (jedes Endgerät im Netzwerk) direkt angesprochen werden kann. Zu diesem Zweck verfügt jeder Knoten über eine IP-Adresse.

IP ist für die Zustellung der Datenpakete verantwortlich, hat jedoch keine Mechanismen zur Fehlerkorrektur. Werden TCP-Datagramme transportiert, stellt TCP sicher, dass auch alle Daten garantiert fehlerfrei übertragen werden. Bei UDP-Datagrammen hingegen steht die fehlerfreie Übertragung zugunsten einer maximalen Performance nicht im Vordergrund.

**Zustellung ohne
Fehlerkorrektur**

IP zerlegt die Datenpakete der darüber liegenden Schicht in IP-Pakete, welche ihrerseits aus dem IP-Header und dem Datenteil bestehen.

Tabelle 2.3:
Aufbau des Headers
eines IP-Paketes

Bezeichnung	Länge	Beschreibung
VERSION	4	IP-Version: 4 = IPv4 6 = IPv6
HLEN (Internet Header Length)	4	Anzahl der 32-Bit-Wörter des Headers
SERVICE TYPE	8	Bits 0-2 haben folgende Bedeutung: 000 – ROUTINE 001 – PRIORITY 010 – IMMEDIATE 011 – FLASH 100 – FLASH OVERRIDE 101 – CRITIC/ECP 110 – INTERNETWORK CONTROL 111 – NETWORK CONTROL Bit 3, DELAY, ist normalerweise Null, für eilige (<i>urgent</i>) Pakete Eins. Bit 4, THROUGHPUT, steuert die Durchleitung, Bit 5, RELIABILITY, die Zuverlässigkeit. Die Bits 6 und 7 werden nicht verwendet.
TOTAL LENGTH	16	Die Länge des gesamten Datagramms einschließlich Daten. Die Länge darf bis zu 65 535 Byte betragen.
IDENTIFICATION	16	Eine vom Absender festgelegte, eindeutige Nummer. Mit Hilfe dieser Nummer werden fragmentierte Datagramme wieder zusammengesetzt.
FRAGMENT FLAGS	3	Bit 0 ist immer 0, Bit 1 steuert die Fragmentierung (0 = Fragmentierung erlaubt, 1 = Fragmentierung verboten). Bit 2 ist 1, wenn weitere Fragmente folgen, und 0, wenn das Datagramm das letzte Fragment ist.
FRAGMENT OFFSET	13	Diese Zahl gibt an, welche Position das Fragment innerhalb des Datagramms hat.

Bezeichnung	Länge	Beschreibung
TTL (Time To Live)	8	Lebensdauer in Hops. Hops sind die Stationen, die das Datagramm durchlaufen kann. Physikalisch ist jeder Router auf dem Weg ein Hop. Jeder Router reduziert den Wert TTL um 1. Ist der Wert 0, wird das Datagramm vernichtet. So wird verhindert, dass Datagramme auf der Suche nach dem Empfänger das Netz unendlich lange durchlaufen.
PROTOCOL	8	Das Protokoll, von dem das Datagramm initiiert wurde: ICMP - Dezimalwert 1 IGMP - Dezimalwert 2 TCP - Dezimalwert 6 EGP - Dezimalwert 8 UDP - Dezimalwert 17 OSPF - Dezimalwert 89
HEADER CHECKSUM	16	Eine Prüfsumme zur Kontrolle der Integrität
SOURCE IP-ADDRESS	32	Die IP-Adresse des Absenders
DESTINATION IP-ADDRESS	32	Die IP-Adresse des Empfängers
IP OPTIONS		0 bis 11 32-Bit-Wörter Optionale Angaben, die nicht fest durch IP spezifiziert sind
PADDING	variabel	Auffüllwert auf ganze Bytes
DATA		Daten

Für den Datentransport im Netzwerk besitzt IP die Fähigkeit, die Pakete in kleinere Einheiten aufzuteilen (fragmentieren). Das kann notwendig sein, wenn das zu übertragene Paket die maximale IP-Paketgrößenbeschränkung eines Netzwerkgerätes (beispielsweise eines IP-Routers) überschreitet. Dieser Parameter wird auch mit MTU (*Maximum Transmission Unit*) bezeichnet.

**IP-Fragmentierung
MTU-Parameter**

Fragmentierte IP-Datenpakete können ein nicht unerhebliches Sicherheitsrisiko darstellen. Die einzelnen Fragmente können manipuliert den Zielhost erreichen. Geschickte Hacker sind in der Lage die Fragmente so zu bilden, dass diese nicht direkt aneinander passen, sondern gemeinsam überlappende Bereiche enthalten. Beim Zusammen setzen im Zielsystem kann es dann durchaus dazu kommen, dass sich

**Sicherheitsrisiko
IP-Fragmente**

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

das Betriebssystem ins Nirwana verabschiedet. Heute gängige Firewall-Systeme weisen IP-Fragmente in der Regel ab.

Path MTU Discovery

Durch den Einsatz der »Path MTU Discovery«-Technologie in Netzwerksystemen wie Routern wird die IP-Fragmentierung überflüssig. Dabei handeln die beteiligten Systeme untereinander aus, wie groß die maximale Paketgröße (MTU) sein darf. Der eine Host startet dann Übertragungsversuche mit steigenden IP-Paketgrößen (Fragmentierungsflag: »Nicht fragmentieren«). Dies geschieht solange, bis er eine ICMP-Fehlermeldung (»Paket zu groß«) zurückerhält.

IP-Fragmentierung wird also im Internet immer seltener, sodass Sie kaum Einschränkungen zu befürchten haben, wenn Sie generell fragmentierte IP-Pakete abweisen. Beachten Sie dabei, wie Sie die Einrichtungsschritte für Ihre Firewall durchführen müssen.

IP-Broadcast

Die meisten IP-Pakete im Netzwerk werden an einen bestimmten Zielknoten geschickt. Dies wird auch mit Unicast bezeichnet. Gehen IP-Pakete an alle erreichbaren Knoten, spricht man von Broadcast. Mit der Broadcast-Adresse 192.168.100.255 erreichen Sie alle Hosts im angenommenen Netzwerk 192.168.100 (Netzmaske 255.255.255.0). Soll eine Nachricht an die über Router verbundenen Netzwerke 192.168.100, 192.168.101 und 192.168.102 gehen, ist die Broadcast-Adresse 192.168.255.255.

IP-Multicast

Über IP-Multicast lassen sich bestimmte Hosts adressieren. Dazu werden Adressen aus dem IP-Bereich 224.0.0.0 bis 239.255.255.255 gewählt und zur Bildung von so genannten Multicast-Gruppen benutzt. Eine einzelne IP-Adresse aus diesem Bereich steht dann für eine Multicast-Gruppe (beispielsweise 224.1.1.22). Über IP-Multicast-Pakete werden nur IP-Protokolle übertragen, die nicht sitzungsorientiert (wie etwa TCP; siehe nächster Abschnitt) arbeiten. Das sind beispielsweise UDP oder Routingprotokolle wie IGMP und OSPF.

Über UDP (siehe ➡ Abschnitt *User Datagram Protocol (UDP)* ab Seite 68) lassen sich Datenströme übertragen, bei denen es auf einen absolut fehlerfreien Transport nicht ankommt. Bei der Verwendung von IP-Multicast anstelle von Unicast lässt sich die verfügbare Bandbreite für eine höhere Anzahl von Nutzern wesentlich effektiver ausnutzen. Statt Einzelverbindungen mit dem entsprechenden Overhead aufzusetzen, können die den entsprechenden Multicast-Gruppen zugewiesenen Hosts den Datenstrom direkt empfangen. Dabei ist die Vorgehensweise mit dem des Abbonierens eines bestimmten Fernsehkabelkanals vergleichbar. Eine Kabelgesellschaft speist eine Reihe von Kanälen in das Kabel ein, die jeweils nur von verschiedenen Gruppen von Kunden empfangen werden können.

Damit eignet sich IP-Multicast insbesondere für die Implementierung von Audio- und Video-Streaming (beispielsweise für Konferenzsysteme). Das ist momentan auch die häufigste Anwendung im Internet.

Die heute gebräuchliche und jedem bekannte Form einer IP-Adresse besteht aus vier dezimalen Zahlen, die jeweils durch einen Punkt voneinander getrennt sind. Hier kann sich in Zukunft einiges ändern, sodass sich eine nähere Betrachtung der IP-Adressversionen lohnt.

**IP-
Adressversionen**

- Internet Protocol Version 4

Im derzeitigen Standard IPv4 (*Internet Protocol Version 4*) besteht die IP-Adresse aus 4 Oktetts. Jedes Oktett entspricht einem Byte (0–255). Zur besseren Lesbarkeit werden sie dezimal ausgeschrieben und durch Punkte getrennt (beispielsweise 195.145.212.138). Theoretisch lassen sich damit $256^4 = 2^{32} = 4\,294\,967\,296$ verschiedene Adressen darstellen. In der Realität verbleiben aber weniger direkt im Internet nutzbare Adressen übrig, da ein Teil davon für die nichtöffentliche Verwendung reserviert ist (siehe auch ➡ Abschnitt *Spezielle IP-Adressen* ab Seite 65). Letztlich bleibt festzustellen, dass der einmal mit IPv4 definierte Adressraum langsam knapp wird und auf absehbare Zeit nicht mehr ausreicht.

Heutiger Standard

- Internet Protocol Version 6

Mit IPv6 wird die Größe einer IP-Adresse von 4 auf 16 Oktetts erweitert. Der derzeitigen Adressenverknappung mit IPv4 kann damit massiv entgegengetreten werden. Es können jetzt 2^{128} statt 2^{32} Adressen gebildet werden. Dies entspricht einer Menge von etwa $3,4 \times 10^{38}$ Computern oder anderen Systemen, die mit einer eindeutigen IP-Adresse versorgt werden könnten, was auch für die weitere Zukunft ausreichend dimensioniert ist.

Zukunft

Neben einer grundsätzlich höheren Anzahl an verfügbaren Adressen bringt IPv6 auch weitere Möglichkeiten mit. So lassen sich beispielsweise unterschiedliche Datentypen spezifizieren (wie etwa Video- oder Ton-Übertragungen), die gegenüber weniger zeitkritischen Datentypen (zum Beispiel E-Mails) bevorzugt bearbeitet werden. Damit können Echtzeitanwendungen besser mit der nötigen Bandbreite ausgeführt werden.

Erweiterte Möglichkeiten

Diese neue IP-Version steht kurz vor der Praxiseinführung. Erste Geräte unterstützen es bereits, der Großteil des Internets läuft aber noch unter der alten Version 4. Alle folgenden Ausführungen im vorliegenden Buch sind der derzeitigen Praxis angepasst und beschränken sich auf die aktuelle IP-Version 4.

Jede IP-Adresse wird in einen *Netzwerk-* und einen *Hostbereich* aufgeteilt. Dafür wird eine so genannte *Subnetzmaske* eingerichtet, die angibt, wie viele Bits einer Adresse zum Netz und wie viele zum Rech-

**Subnetze und
Netzwerkklassen**

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

ner gehören. Hier ein Beispiel in dezimaler und binärer Notation für die IP-Adresse 192.168.100.38 mit der Subnetzmaske 255.255.255.0.

Tabelle 2.4:
Netzwerk- und
Hostadresse in
dezimaler und
binärer Form

	Netzwerkbereich		Hostbereich	
	Dezimal	Binär	Dez	Binär
Subnetz- maske	255.255.255	11111111.11111111.11111111	0	00000000
IP-Adresse	192.168.100	11000000.10101000.01100100	38	00100110

Mit der Subnetzmaske 255.255.255.0 können in einem Netzwerk bis zu 254 Rechnern adressiert werden. Das ist für kleinere Netzumgebungen ausreichend. Die Null ist als reguläre Host-Adresse nicht zulässig (kennzeichnet das Netzwerk), ebenso die 255. Die 255 wird als Broadcast-Adresse benutzt, wenn alle Hosts angesprochen werden sollen (siehe auch ➡ Seite 62).

Subnetzmaske

Die Subnetzmaske besteht generell aus einem durchgängigen Bereich von binären Einsen. Es hat sich eingebürgert, die Einsen zu zählen und in der Kurzform /n hinter der Netzwerkadresse aufzuschreiben. Eine Angabe von 192.168.100.0/24 bedeutet also Netzadressen im Bereich von 192.168.100.x mit einer Subnetzmaske von 255.255.255.0 (24 Einsen).

Über die Aufsplittung der IP-Adresse in den Netzwerk- und den Hostbereich kann der Host einfach feststellen, ob diese im eigenen (Sub-)Netz oder in einem anderen liegt. In unserem Beispiel würde dann die Adresse 192.168.101.56 einen Host im (anderen) Subnetz 192.168.101 adressieren, während 192.168.100.78 im gleichen Netz zu finden ist.

Netzwerkklassen

Eine IP-Adresse enthält im Netzwerkbereich eine Netzwerkennung, welche die verwendete Netzwerkklasse angibt. Es werden fünf verschiedene Netzwerkklassen (A bis E) unterschieden, wobei jeder Klasse eine bestimmte Standard-Subnetzmaske zugeordnet ist.

Klasse A

Ein Klasse-A-Netz hat standardmäßig die Subnetzmaske 255.0.0.0. Das erste Bit der Adresse ist auf 0 gesetzt.

128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1
8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1
0	Netzwerk							Host																							

Klasse B

Ein Klasse-B-Netz hat die Subnetzmaske 255.255.0.0. Die ersten beiden Bits der Adresse sind auf 10 gesetzt.

128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1
8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1
0	Netzwerk							Host																							

Ein Klasse-C-Netz hat die Subnetzmaske 255.255.255.0. Die ersten drei Bits der Adresse sind hier auf 110 gesetzt.

Klasse C

128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1
8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1
1	1	0	Netzwerk																		Host										

Daneben gibt es noch Klasse-D- (beginnt mit 1110) und Klasse-E-Netze (beginnend mit 1111). Klasse-D-Adressen dienen zur Bildung von Multicast-Gruppen (siehe Seite 62), Klasse-E-Netze sind für Spezialfälle reserviert.

Klasse D und E

Spezielle IP-Adressen

Es gibt eine Reihe von IP-Adressen, die nicht im öffentlichen Internet oder generell nicht im Netzwerk selbst zum Einsatz kommen und für spezielle Einsatzzwecke reserviert sind.

Eine *Broadcast-Adresse* teilt dem Rechner mit, wie er alle Rechner in seinem Netz auf einmal erreichen kann (sog. *Broadcast*). Dabei werden einfach alle Bits im Rechnerbereich der Adresse auf 1 gesetzt (allgemeingültige Definition für *ALL-ONE-Broadcasts*). Die Standard-Broadcast-Adresse für einen Rechner aus dem Netz 192.168.100.0/24 wäre demnach 192.168.100.255. Sie können deshalb Adressen, die auf 255 enden, nicht als reguläre Netzwerkadresse angeben.

Broadcast-Adressen

Mit einer Adresse, die im ersten Oktett eine 127 enthält, adressiert sich jeder Rechner selbst (*Loopback*), was zu Tests der Netzwerksoftware benutzt werden kann. Eine solche Adresse ist daher niemals auf dem Kabel zu sehen.

Loopback

Adressen aus den Klasse-D- und -E-Netzen sind für bestimmte Zwecke reserviert. Die Adressen 224.x.x.x bis 255.x.x.x dürfen deshalb nicht für die normale Adressierung von Hosts benutzt werden. Genauere Informationen dazu stehen im RFC 2236.

Reservierte Adressen

Private Netzwerkadressen

In jeder IP-Netzklasse (siehe vorhergehender Abschnitt) gibt es Adressbereiche, die nicht im Internet selbst zulässig sind und somit für die Implementierung lokaler Netzwerke genutzt werden können.

Tabelle 2.5:
Private Netzwerk-
adressen je Netz-
klasse

Klasse	Anz. Subnetze	Nutzbare Adressbereiche
A	1	10.0.0.0 bis 10.0.0.255
B	16	172.16.0.0 bis 172.31.255.255
C	256	192.168.0.0 bis 192.168.255.255

NAT

Für die Anbindung lokaler Netzwerke an das Internet, in denen diese privaten IP-Adressen verwendet werden, kommt NAT (*Network Address Translation*) zum Einsatz. Dabei werden die Anfragen der Clients, die über eine private IP verfügen, in die jeweilige öffentliche IP-Adresse des Internet-Routers übersetzt. Dieses Verfahren wird auch *Masquerading* genannt. NAT kommt beispielsweise in Internet-Routern zum Einsatz, die lokale Netzwerke mit dem Internet verbinden.

IP-Adressvergabe im Internet

Jede öffentliche IP-Adresse ist weltweit eindeutig und wird von der IANA an die drei Organisationen APNIC, ARIN und RIPE vergeben, die diese dann wiederum an Endkunden (Firmen oder Internetprovider) verteilen. Weitere Informationen gibt es bei den entsprechenden Organisationen unter folgenden Adressen:

- IANA (Internet Assigned Numbers Authority):

www.iana.net

- APNIC (Asia-Pacific Network Information Center):

www.apnic.net

- ARIN (American Registry for Internet Numbers):

www.arin.net

- RIPE NCC (Réseaux IP Européens):

www.ripe.net

Generell bleibt festzuhalten, dass jegliche Verwendung von IP-Adressen bei direkt am Internet angeschlossenen Computern oder anderen Netzwerkgeräten sich nach diesen Bestimmungen zu richten hat. Für den Aufbau lokaler Netzwerke empfiehlt sich im Regelfall die Einrichtung von IP-Adressen aus dem nichtöffentlichen (privaten) Adressbereich (siehe vorhergehender Abschnitt).

Transmission Control Protocol (TCP)

Verbindungsorientiert mit Fehlerkorrektur

Dieses Protokoll ist das meistbenutzte der Schicht 3 (Transport) der Internet-Protokollfamilie. Es arbeitet verbindungsorientiert und ist in der Lage, eine Fehlerkorrektur durchzuführen. Eine Verbindung wird dabei über Ports zwischen Sender und Empfänger hergestellt (siehe auch ➡ Abschnitt *Port- und Protokollnummern* ab Seite 69). Damit ist

auch ein gleichzeitiges Senden und Empfangen, eine so genannte *voll-duplexe Verbindung*, möglich.

Feld	Länge	Beschreibung
SOURCE PORT	16	TCP-Quellport
DEST PORT	16	TCP-Zielpport
SEQUENZ NR.	32	Sequenznummer
ACKN. NR.	32	Bestätigungsnummer
DATA OFFSET	4	Anzahl der 32-Bit Wörter im TCP-Vorspann
RESERVED	6	Reserviert
FLAGS	6	6 Flags: URG - Dringende Übertragung ACK - Bestätigung (ACKN. NR. ist gültig) PSH - Push, Daten werden sofort an die höhere Schicht weitergegeben RST - Reset, Verbindung wird zurückgesetzt SYN - Sync-Flag; dient zusammen mit ACK zum Aufbau der TCP-Verbindung FIN - Finale-Flag; beendet die Verbindung
WINDOW	16	Dient der Flusststeuerung
CHECKSUM	16	Prüfsumme
URGENT PTR	16	Ist gültig, wenn das URG-Flag gesetzt ist und zeigt auf die Folgenummer des letzten Bytes des Datenstroms.
OPTIONS	max. 40	Optionaler Teil
PADDING		Füllzeichen, um auf volle 32-Bit zu kommen
DATA		Daten

Tabelle 2.6:
Aufbau eines TCP-Pakets

Für den Aufbau einer TCP-Verbindung spielen das ACK- und das SYN-Flag eine entscheidende Rolle. So ist beim ersten TCP-Paket das ACK-Flag stets auf 0 gesetzt. Mit einem Handshake über drei Datenpakete wird die Verbindung aufgebaut.

Aufbau einer TCP-Verbindung

Zum Beenden der Verbindung werden das RST- oder das FIN-Flag benutzt. Ein gesetztes RST zeigt einen Verbindungsfehler an, während über FIN (wird sowohl von Empfänger als auch vom Sender im je-

Beenden der TCP-Verbindung

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

weils letzten Paket gesetzt) ein normaler Verbindungsabbau durchgeführt wird.

Kontrolle der Paketreihenfolge

Über die Sequenz- und Bestätigungsnummern wird dafür gesorgt, dass alle Datenpakete in der richtigen Reihenfolge beim Empfänger zusammengesetzt und doppelt versandte Pakete ignoriert werden können. Beide Hosts generieren unabhängig voneinander eine eigenständige Sequenznummer, die sie sich beim Aufbau der Verbindung übermitteln (wenn SYN gesetzt ist). Danach werden die Sequenznummern jeweils erhöht (um die Anzahl der Datenbytes im Paket). Damit wird sichergestellt, dass die Pakete beim Empfänger in der richtigen Reihenfolge wieder zusammengesetzt werden können.

Prüfsumme

Für die Sicherstellung eines ordnungsgemäßen Datentransfers ist allein die Kontrolle der richtigen Reihenfolge der Pakete nicht ausreichend. Über die Prüfsumme kann daher ermittelt werden, ob das Datenpaket selbst korrekt übertragen worden ist. Die Prüfsumme wird aus der Summe der 16-Bit-Wörter des TCP-Pakets berechnet, wobei bestimmte IP-Headerinformationen mit einbezogen werden.

User Datagram Protocol (UDP)**User Datagram Protocol**

Das Protokoll UDP arbeitet, anders als TCP, nicht verbindungsorientiert (»verbindungslos«) und besitzt keine Kontrollmöglichkeit, um die Reihenfolge von UDP-Paketen beziehungsweise die Vollständigkeit eines UDP-Datenstroms zu sichern. Allerdings ist eine einfache Fehlerprüfung der einzelnen Pakete über eine Prüfsumme möglich.

Verbindungsloses Protokoll

Damit eignet sich UDP hervorragend für Anwendungen, die eine direkte Verbindung zwischen Sender und Empfänger nicht benötigen. Der Overhead, der beim Auf- und Abbau der Verbindung wie bei TCP entsteht, entfällt und eine hohe Performance wird erreichbar. Die wird beispielsweise bei Streaming-Video-Anwendungen benötigt. Die Priorität ist dabei so gesetzt, dass es vor allem darauf ankommt, dass der Empfänger überhaupt ein fortlaufendes Bild erhält. Gehen vereinzelt Daten verloren, wird es vielleicht Bildstörungen geben. Der Informationsinhalt wird trotzdem übertragen.

DNS-Abfragen über UDP

Ein anderes Beispiel stellen Nameserver-Abfragen dar. Diese werden ebenfalls über UDP abgewickelt. Bei der Vielzahl der üblicherweise notwendigen Abfragen über kleine Datenpakete wird damit eine optimale Performance erreicht. Kommt von einem Nameserver keine Antwort, wird einfach der nächste Server kontaktiert. Theoretisch können DNS-Serverdienste auch über TCP abgewickelt werden. Allerdings hängt dies von der jeweiligen Implementierung ab. Der Windows 2000 DNS-Server unterstützt dies nicht.

Feld	Länge	Beschreibung
SOURCE PORT	16	UDP-Quellport
DEST PORT	16	UDP-Zielpport
LENGTH	16	Länge des UDP-Pakets in Bytes (Header plus Daten)
CHECKSUM	16	Prüfsummenfeld
DATA		Daten

Tabelle 2.7:
Aufbau eines UDP-Paketes

UDP hat wie beschrieben keine Möglichkeiten zur Flusskontrolle. Allerdings kann über die UDP-Prüfsumme ermittelt werden, ob das Datenpaket selbst korrekt übertragen worden ist. Die Prüfsumme wird aus den Werten des UDP-Pakets unter Einbeziehung bestimmter IP-Headerinformationen berechnet.

Prüfsumme

Unter anderem verwenden die folgenden Anwendungen das Protokoll UDP:

UDP-Anwendungen

- DNS (*Domain Name System*)
- NFS (*Network File System*; nur unter Unix bedeutsam)
- RIP (*Routing Information Protocol*)
- SNMP (*Simple Network Management Protocol*)
- TFTP (*Trivial File Transfer Protocol*)

Zu beachten ist, dass UDP kein sicheres Protokoll ist. Aufgrund der nicht vorhandenen Flusskontrolle können in einem Datenstrom leicht UDP-Pakete gefälscht oder gefälschte UDP-Pakete eingeschmuggelt werden. Auch lassen sich wirksame Denial of Service-Attacken gegen Hosts fahren, indem diese mit UDP-Paketen überflutet werden.

Unsicheres Protokoll

Port- und Protokollnummern

Für die eindeutige Identifizierung der Protokolle und Ports bei der Netzwirkkommunikation über IP, TCP und UDP gibt es die so genannten Port- und Protokollnummern. Vor der Explosion der Protokolle (es gibt inzwischen Hunderte solcher Kombinationen aus Protokollen und Ports), wurden diese in der RFC 1700 geführt. Da RFCs keine Versionsnummer besitzen und bei jeder Änderung durch eine neue ersetzt werden, würde dies zu einer Inflation von RFCs führen. Die für die Nummernvergabe zuständige Organisation IANA verwaltet deshalb die Nummern heute direkt auf ihrer Website:

<http://www.iana.org>

Ports

Multiplexing

Damit ein Rechner gleichzeitig mehrere Verbindungen (*Multiplexing*) bearbeiten kann, müssen diese unterschieden werden. Dazu bedient sich das TCP der *Ports*. Jeder Anwendung, die das TCP benutzen will, wird ein Port zugeordnet. Es gibt 65 535 verschiedene Ports, fortlaufend nummeriert. Dabei gelten folgende Grundsätze:

- Ein Paar aus IP-Adresse und Port wird *Socket* genannt.
- Eine Verbindung zwischen zwei Rechnern ist wiederum eindeutig durch *zwei Sockets* definiert.
- Ein Rechner kann mehrere TCP-Verbindungen gleichzeitig bearbeiten. Dafür werden verschiedene Ports definiert. Dieser Vorgang wird als »Multiplexing« bezeichnet.

Eine Portbezeichnung wird normalerweise hinter einem Doppelpunkt an die IP-Adresse oder den DNS-Namen gehängt, beispielsweise wie folgt: 192.168.0.101:80.

Ports

Das *Port-Konzept* lässt sich in etwa mit einer Telefonnummer vergleichen: Der Netzwerkteil einer Internet-Adresse entspricht der Vorwahl, der Host-Teil der eigentlichen Telefonnummer und der Port schließlich einer Nebenstellenummer. Dabei wird eine TCP-Verbindung generell eindeutig durch die beteiligten Sockets definiert (Sender und Empfänger). Es kann keine zwei identischen Socket-Paare zur gleichen Zeit geben. Der Sender bestimmt eine Portnummer per Zufallsgenerator. Damit ist es beispielsweise möglich, dass von einem Rechner zwei Telnet-Verbindungen zu dem gleichen Zielrechner existieren. In einem solchen Fall unterscheiden sich dann jedoch die einzelnen Portnummern des Client-Rechners. Beim Verbindungsaufbau leitet die Anwendungsschicht das Datenpaket mit der Internet-Adresse des Servers und dem Port 21 an die Transportschicht weiter. Da TCP stromorientiert sendet, verläuft die Übertragung der Bytes in der gleichen Reihenfolge vom Client zum Server und vermittelt der Anwendungsschicht das Bild eines kontinuierlichen Datenstroms.

Auf den meisten Systemen sind die Ports über 1 024 für jede Anwendung offen, während die Ports 1 – 1 024 nur Systemprozessen (oder Anwendungen, die über entsprechende Privilegien verfügen) zur Verfügung stehen. Die folgende Tabelle zeigt die wichtigsten Ports.

Tabelle 2.8:
Einige wichtige
Portnummern

Dienst	Port	Erklärung
ftp-data	20	File Transfer [Default Data]
ftp	21	File Transfer [Control]
telnet	23	Telnet
smtp	25	Simple Mail Transfer

Dienst	Port	Erklärung
domain	53	Domain Name Server
finger	79	Finger
www-http	80	World Wide Web HTTP
pop3	110	Post Office Protocol – Version 3
uucp-path	117	UUCP Path Service
nntp	119	Network News Transfer Protocol
ntp	123	Network Time Protocol
netbios-ns	137	NETBIOS Name Service
netbios-dgm	138	NETBIOS Datagram Service
netbios-ssn	139	NETBIOS Session Service
imap2	143	Interim Mail Access Protocol v2
Irc	194	Internet Relay Chat Protocol
Ipx	213	IPX
imap3	220	Interactive Mail Access Protocol v3
uucp	540	uucpd

Socket ist ein im Zusammenhang mit TCP/IP häufig verwendeter Begriff, der die Kombination aus Internet-Adresse und Portnummer bezeichnet. Innerhalb der Transportschicht werden bestimmte Ports zur Adressierung verwendet. Sowohl UDP als auch TCP verwenden Port-Adressen, um Daten an das betreffende Programm (Protokoll) der Anwendungsschicht zu senden, wobei beide teilweise unterschiedliche Dienste für die gleiche Portnummer vermitteln.

Socket

Protokollnummern

Im Feld Header des IP-Datenpakets (siehe auch Tabelle 2.3 auf Seite 60) finden Sie die Nummer des nächsthöheren Protokolls, an das die Daten weitergeleitet werden sollen. Diese Nummern sind für alle Protokolle der Internet-Protokollfamilie definiert.

Die folgende Abbildung zeigt als Beispiel eine Datei PROTOCOL eines Windows 2000 Systems. In ähnlicher Form ist dies auch unter Linux zu finden.

```
# <Protokollname> <Nummer> [Alias...] [#<Kommentar>]
ip      0      IP      # Internet Protocol
icmp    1      ICMP    # Internet Control Message Protocol
ggp     3      GGP     # Gateway-Gateway Protocol
tcp     6      TCP     # Transmission Control Protocol
```

Listing 2.1:
Inhalt der Datei
PROTOCOL

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E


```

egp      8      EGP      # Exterior Gateway Protocol
pup      12     PUP      # PARC Universal Packet Protocol
udp      17     UDP      # User Datagram Protocol
hmp      20     HMP      # Host Monitoring Protocol
xns-idp  22     XNS-IDP  # Xerox NS IDP
rdp      27     RDP      # "Reliable Datagram" Protocol
rvd      66     RVD      # MIT Remote Virtual Disk

```

Diese Datei ist eine normale ASCII-Textdatei und kann mit dem Editor geöffnet werden.

2.2 Höhere Netzwerkprotokolle

Die in den folgenden Abschnitten behandelten Protokolle arbeiten auf Ebene der Anwendungsschicht (Schicht 4 der Internet-Protokollfamilie). Dabei werden hier die Protokolle näher behandelt, die im Zusammenhang mit dem Einsatz von Webdiensten eine besondere Rolle spielen.

Tabelle 2.9:
Übersicht über die
behandelten höheren
Protokolle

Protokoll	Bezeichnung	Seite
HTTP	Hypertext Transfer Protocol	72
FTP	File Transfer Protocol	77
SMTP	Simple Mail Transfer Protocol	84
NNTP	Network News Transfer Protocol	87

Weitere Hinweise zum Schichtenmodell der Internet-Protokollfamilie finden Sie auch im ➡ Abschnitt 2.1.6 *Das ISO/OSI-Modell und die Internet-Protokolle* ab Seite 54.

Die hier vorgestellten Protokolle der höheren Schichten arbeiten zeilenorientiert. Das Ende eines Kommandos wird also spätestens durch das Senden eines Zeilenvorschubs angezeigt. Der normale Ablauf geht davon aus, dass der Server an dem entsprechenden Port auf Kommandos hört (er lauscht). Trifft ein Kommando ein, interpretiert er es und sendet seinerseits eine entsprechende Reaktion. Manche Protokolle, wie beispielsweise FTP, interpretieren schon den erfolgreichen Verbindungsaufbau als Kommando – der Server reagiert dann sofort mit einer Begrüßung.

2.2.1 HTTP

HTTP (*Hypertext Transfer Protocol*) dient der Kommunikation mit Webservern. Es gibt derzeit zwei Versionen, 1.0 und 1.1. Das modernere 1.1 steht allerdings nicht allen Servern zur Verfügung. Auf Seiten der Browser dominiert inzwischen HTTP 1.1, denn alle Browser ab Versi-

on 4 beherrschen dieses Protokoll. Der Internet Information Server 5 beherrscht die Version 1.1 vollständig.

HTTP 1.0 wurde im Mai 1996 in der RFC 1945 veröffentlicht, schon im August desselben Jahres folgte HTTP 1.1. Für das neuere Protokoll stand auch in 1999 noch kein eigenes RFC zur Verfügung, nur ein Draft. Trotz der langen Zeit (für Internet-Verhältnisse) und den enormen Vorteilen von HTTP 1.x sind immer noch Server mit der Entwicklungsversion 0.9 im Einsatz. **RFC 1945**

Bei HTTP handelt es sich, wie bei Finger, Telnet und Echo auch, um ein verbindungs- oder statusloses Protokoll. Server und Client nehmen also nie einen besonderen Zustand ein, sondern beenden nach jedem Kommando den Prozess komplett, entweder mit Erfolg oder mit einer Fehlermeldung. Es obliegt dem Kommunikationspartner, darauf in angemessener Weise zu reagieren. **Verbindungsloses Protokoll**

Protokollaufbau

HTTP-Kommandos können aus mehreren Zeilen bestehen. Die erste Zeile ist immer die Kommandozeile. Daran angehängt kann ein Message-Header folgen. Der Header enthält weitere Parameter, die das Kommando spezifizieren. So kann ein Content-Length-Feld enthalten sein. Steht dort ein Wert größer als 0, folgen dem Header Daten. Die Daten werden also gleich zusammen mit dem Kommando gesendet, man spricht dann vom Body der Nachricht. HTTP versteht im Gegensatz zu SMTP den Umgang mit 8-Bit-Werten. Binärdaten, wie Bilder oder Sounds, müssen nicht konvertiert werden. **Header Body**

Folgen dem HTTP-Kommando und den Header-Zeilen zwei Leerzeilen (Zeilenwechsel »\n«), so gilt das Kommando als beendet. Kommandos mit Body haben kein spezielles Ende-Zeichen, das Content-Length-Feld bestimmt, wie viele Bytes als Inhalt der Nachricht eingelesen werden.

Ein HTTP-Kommando hat immer folgenden Aufbau:

```
METHODE ID VERSION
```

Mit *METHODE* wird das Kommando selbst bezeichnet. Die folgende Tabelle zeigt die HTTP-Kommandos auf einen Blick. **Aufbau eines HTTP-Kommandos**

Name	Beschreibung
DELETE	Ressource löschen
GET	Ressource anfordern
HEAD	Header der Ressource anfordern
LINK	Verknüpfung zweier Ressourcen beantragen

Tabelle 2.10:
Die HTTP-Kommandos

Name	Beschreibung
OPTIONS	Optionen des Webserver erfragen
POST	Daten an einen Serverprozess senden
PUT	Ressource auf dem Webserver ablegen
TRACE	Kommando zurückschicken lassen
UNLINK	Verknüpfung zwischen Ressourcen löschen

Beachten Sie, dass die Kommandos unbedingt in Großbuchstaben geschrieben werden müssen, exakt wie in Tabelle 2.10 gezeigt. Die ID einer Ressource kann beispielsweise eine Adresse oder ein Dateiname sein:

```
GET index.htm HTTP/1.0
```

Dieses Kommando fordert die Datei INDEX.HTM an.

Statuscodes

Die Antwort auf ein Kommando besteht im Senden eines Statuscodes. Dem Statuscode folgen optionale Felder und, bei der Übertragung von Ressourcen, die Daten. Die Statuszeile hat folgenden Aufbau:

```
VERSION STATUSCODE STATUSTEXT
```

Der Statuscode ist eine dreistellige Ziffer, von denen die erste (Hunderter) die Zuordnung zu einer bestimmten Gruppe zeigt. In der Referenz finden Sie eine ausführliche Beschreibung aller Statusmeldungen; Tabelle 2.11 zeigt eine Übersicht der wichtigsten.

Tabelle 2.11:
Statuscodes einer
HTTP-Antwort

Statuscode	Beschreibung
200	Kommando erfolgreich
201	Ressource wurde erstellt
202	Authentifizierung akzeptiert
204	Kein Inhalt oder nicht angefordert
301	Ressource am anderen Ort
302	Ressource nicht verfügbar (temporär Zustand)
304	Ressource wurde nicht verändert (steuert Proxy)
400	Syntaxfehler
401	Keine Autorisierung
403	Nicht öffentlicher Bereich
404	Nicht gefunden (der berühmteste HTTP-Fehler!)

Statuscode	Beschreibung
500	Serverfehler, Fehlfunktion
501	Kommando nicht implementiert
502	Feldwert oder URL ungültig (nur Proxy)
503	Dienst nicht verfügbar

Die Einteilung in Gruppen genügt oft für die Programmierung:

- 2xx. Kommando erfolgreich
- 3xx. Weitere Reaktion erforderlich
- 4xx. Fehler, Wiederholung mit anderen Daten sinnvoll
- 5xx. Serverfehler, Wiederholung zwecklos

Der HTTP-Message-Header

An ein Kommando oder an die Statuszeile können weitere Felder angehängt werden. Der Aufbau lehnt an den MIME-Standard an. Die Header-Felder können in drei Hauptgruppen aufgeteilt werden:

- F. Frage-Felder (Request-Header-Fields) sind nur in Kommandos erlaubt.
- A. Antwort-Felder (Response-Header-Fields) kommen nur in der Antwort (Statusnachricht) vor.
- I. Informationsfelder (General-Header-Fields) übertragen alle anderen Nachrichten, wie Größen und Parameter.

Nicht alle Server stellen alle Felder zur Verfügung, teilweise ergeben sich durch die Weiterleitung der Nachrichten an den Nutzer – immerhin empfängt die Daten der Browser – eine erhebliche Sicherheitslücke. Wenn ein Feld mehrfach übertragen werden muss, kann die Angabe der Werte als kommaseparierte Liste oder durch Wiederholung der Feldnamen erfolgen:

```
Header-Field Wert, Wert, Wert ..
```

oder

```
Header-Field Wert
Header-Field Wert
Header-Field Wert
```

Einige Header können untergeordnete (optionale) Informationen enthalten. So kann dem Content-Type der Name der Datei übergeben werden. Diese Elemente werden durch ein Semikolon getrennt:

```
Content-Type: application/pdf; name=orderform.pdf
```

V

S

I

1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E

In der folgenden Tabelle finden Sie alle Header-Felder und die zugehörigen Gruppen (»Grp« in der folgenden Tabelle).

Tabelle 2.12:
Header-Felder und
Feldgruppen

Grp	Feldname	Beschreibung
F	Accept	MIME-Typen, die der Client verarbeiten kann
F	Accept-Charset	Bevorzugter Zeichensatz
F	Accept-Encoding	Kodierung des Clients
F	Accept-Language	Sprache des Clients
I	Allow	Liste aller erlaubten Kommandos
F	Authorization	Authentifizierung des Clients
I	Content-Disposition	Inhaltsbeschreibung einer MIME-Quelle
I	Content-Encoding	Kodierung der Ressource
I	Content-Language	Sprache der Ressource
I	Content-Length	Größe der Ressource (in Byte)
I	Content-Type	MIME-Typ der Ressource
I	Date	Absende- oder Erstellungsdatum
I	Expires	Verfallsdatum der Ressource
F	From	E-Mail-Adresse des Nutzers
F	Host	Domainname des Webservers
F	If-Modified-Since	Nur dann GET, wenn neueren Datums
I	Last-Modified	Aktualisierungsdatum der Ressource
I	Link	Verknüpfung
A	Location	URL der Ressource (bei Redirect)
I	MIME-Version	MIME-Version des Headers
I	Pragma	Allgemeiner Schalter »Name=Wert«
F	Referer	URL der Herkunfts-Ressource
A	Retry-After	Datum der nächsten Verfügbarkeit
A	Server	Name und Version des Webservers
I	Title	Titel der Ressource
U	URI	URI der Ressource
F	User-Agent	Name und Versionsnummer des Browsers

Grp	Feldname	Beschreibung
A	WWW-Authenticate	Authentifizierungs-Schema

Im Gegensatz zu anderen Protokollen ist die Länge eines Datenblocks im Content-Length festgelegt, irgendwelche Begrenzungszeichen gibt es nicht. Beachtenswert ist auch, dass der Server nach dem Verbindungsaufbau keine Antwort sendet. Erst das erste eintreffende Kommando löst eine Reaktion aus. Darin ist die Ursache zu sehen, wenn die Browser nach der Anforderung eines unerreichbaren Server lange Zeit nicht reagieren. Als »Totsignal« wird einfach eine vorgegebene Zeit gewartet, in welcher der Server auf das erste Kommando reagieren sollte.

Eine einfache HTTP-Verbindung könnte also folgendermaßen aussehen:

```
Client: (Verbindungsaufbau des Browsers)
Server: (keine Antwort)
Client: GET /index.htm HTTP/1.0
      If-Modified-Since: Wed, 30 Jul 1997 00:00:00
      <CRLF>
Server: HTTP/1.0 200 Document follows
      Date: Mon, 18 Aug 1997 23:45:21 GMT+200
      Server: IIS 5.0, Microsoft Corporation
      Content-Type: text/html
      Last-Modified: Mon, 11 Aug 1997 14:43:13
      Content-Length: 7856
      <CRLF>
      JDGGF/(&$=$(?ED`D`?I`... Daten ohne Endeckennung
```

Wenn Header selbst ausgewertet werden, muss das Datumsformat beachtet werden:

```
www, dd MMM YYYY HH:mm:ss GMT+xxx
```

2.2.2 File Transfer Protocol (FTP)

Nach HTTP ist FTP eines der wichtigsten Internet-Protokolle. Mit FTP haben Sie Zugriff auf Teile des Dateisystems eines Servers. FTP wurde in der RFC 959 definiert und stammt von den Vorläufern TFTP (*Trivial File Transfer Protocol*, RFC 1350) und SFTP (*Simple File Transfer Protocol*, RFC 913) ab. TFTP ist kaum noch gebräuchlich, da es sich auf UDP stützt und nicht sicher ist (siehe auch ➡ Abschnitt *User Datagram Protocol (UDP)* ab Seite 68). In der Praxis kommen sie noch bei bestimmten Bootstrap-Protokollen zum Einsatz, die zum Laden von Betriebssystemen über das Netzwerk (so genanntes *Remote Boot*) verwendet werden.

FTP kennt eine Vielzahl von Kommandos. Einige grafische FTP-Clients zeigen diese an, wenn die Kommunikation abläuft. Es ist auch

RFC 959
RFC 1350
RFC 913

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

durchaus gebräuchlich, FTP-Kommandos direkt an der Konsole einzugeben. Auch FTP ist verbindungslos und jedes Kommando umfasst nur eine Zeile. Tabelle 2.13 zeigt einen Überblick über alle einsetzbaren Kommandos.

Tabelle 2.13:
FTP-Kommandos

Kommando	Parameter	Beschreibung
ABOR		Transfer abbrechen
ACCT	Kennung	Zugangskennung
ALLO	Dateigröße	Platz auf dem Server beantragen
APPE	Dateiname	Datei an vorhandene anhängen
CDUP		Eine Verzeichnisebene höher
CWD	Verzeichnis	Verzeichnis wechseln
DELE	Dateiname	Datei löschen
HELP	Kommando	Hilfe anfordern
LIST	[Verzeichnis]	Liste im Verzeichnis anzeigen
MKD	Verzeichnis	Verzeichnis erstellen
MODE	Modus	Datentransfer-Modus festlegen
NLST	[Verzeichnis]	Einfache Dateiliste
NOOP		Verbindung prüfen
PASS	Kennwort	Kennwort des Nutzers
PASV		Passiver Datentransfer-Modus
PORT	Port	Adresse und Port festlegen
PWD		aktuelles Verzeichnis abfragen
QUIT		Verbindung beenden
REIN		Verbindung neu initialisieren
REST	Kennung	Abgebrochenen Transfer neu starten
RETR	Dateiname	Datei von FTP-Server holen
RMD	Verzeichnis	Verzeichnis löschen
RNFR	Dateiname	Datei umbenennen (siehe RNT0)
RNT0	Dateiname	Neuer Name der Datei
STAT	[Dateiname]	Verbindungsstatus abfragen
STOR	Dateiname	Datei ablegen
STOU		Datei mit eindeutigen Namen ablegen

Kommando	Parameter	Beschreibung
STRU	Struktur	Dateistruktur festlegen (Datei, Datensatz oder Seite)
SYST		Betriebssystem des FTP-Servers
TYPE	Typ	Transfer-Typ (ASCII, EBCDIC,...)
USER	Name	Nutzername zur Authentifizierung

Authentifizierung mit FTP

Die Authentifizierung ist auf mehreren Wegen möglich. Sicher kennen Sie selbst FTP-Server, die Name und Kennwort verlangen, während andere den anonymen Zugriff erlauben. Für die Anmeldung an einem geschützten Server sind die Kommandos USER, PASS und optional ACCT zuständig. Die Übertragung der Kennwörter erfolgt generell unverschlüsselt.

Die unverschlüsselte Übertragung von Kennwörtern bei FTP stellt ein erhebliches Sicherheitsrisiko dar. FTP-Server sollten deshalb nur für nicht besonders schützenswerte Informationen, beispielsweise öffentlich zugängliche Datenbestände, eingesetzt werden.



Für öffentlich zugängliche Daten wird meist ein anonymer FTP-Zugang eingerichtet, der ohne weitere Authentifizierung genutzt werden kann. Dabei ist nur eine bestimmte Konvention für Name und Kennwort einzuhalten, die heute auf fast allen FTP-Serversystemen auf die gleiche Art und Weise implementiert ist.

Anonymous-FTP

Mit dem folgenden Befehl wird der Wunsch nach einer anonymen Verbindung mitgeteilt:

```
USER anonymous
```

Das Wort »anonymous« muss exakt in dieser Schreibweise, mit Kleinbuchstaben, geschrieben werden. Beachten Sie auch, dass alle Kommandos mit Großbuchstaben geschrieben werden müssen. Viele FTP-Clients setzen dies allerdings intern um, sodass der Eindruck entsteht, man könne auch mit Kleinbuchstaben arbeiten. Wenn Sie selbst mit einem FTP-Client entwerfen, beispielsweise mit ASP, müssen Sie diese Regeln aber kennen. Auch die anonyme Anmeldung verlangt ein Kennwort. Mit folgendem Befehle senden Sie als Kennwort die eigene E-Mail-Adresse:

```
PASS name@mail.com
```

Ob die Adresse korrekt ist oder nicht spielt keine Rolle, es ergeben sich keine Konsequenzen daraus. Der Server schaltet nun die für anonyme Besucher zulässigen Ressourcen frei. Normalerweise werden nur be-

stimmte Verzeichnisse zum Download freigegeben und grundsätzlich keine Schreibrechte erteilt. Hasardeure mögen dies anders handhaben.

Datenverbindung

Vielleicht haben Sie schon in Listen mit Portnummern neben FTP auch die Zahl 23 bemerkt. FTP benutzt einen Kanal für die Authentifizierung und Steuerung. Dieser Kanal arbeitet normalerweise auf Port 21. Die Übertragung der Daten findet dann auf einem Datenkanal statt, dem Port 23. Der Sinn ist in der Verbindung zweier FTP-Server zu suchen. Wenn Sie einen Datenabgleich zwischen zwei Servern herstellen, muss ein Server den anderen anrufen. Lauschen aber beide auf Port 21, können entweder nur Daten oder nur Kommandos ausgetauscht werden. Durch den zweiten Port bleibt auch während einer langen Datenübertragung der Austausch von Kommandos möglich.

Transfer-Typen

Ein wichtiger Parameter ist die Übertragung des Transfer-Typs. Damit wird das Datenformat festgelegt, in dem die Übertragung der Daten erfolgt. Tabelle 2.14 zeigt die Typen im Detail.

Tabelle 2.14:
Datentransfer-Typen
für FTP

Kürzel	Option	Beschreibung
A	N T I	ASCII, Non-Print, TelNet, Carriage Control
E	N T I	EBCDIC, Non-Print, TelNet, Carriage Control
I		binär, 8-Bit
L	n	binär, n Bit

Zwischen dem Transfer-Typ und der Option muss ein Leerzeichen stehen. Für den normalen Einsatz genügt das Umschalten zwischen A und I. Wenn Sie alle Dateien mit I übertragen, gibt es am wenigsten Probleme.



Hinweis

Die Option A überträgt bei den meisten Servern nur 7-Bit-ASCII, so dass Binärdateien völlig verstümmelt werden. Dazu gehören aber auch Textdateien aus einer Textverarbeitung wie Word, die für ihre Steuerzeichen den gesamten Zeichensatz verwenden. Standardmäßig steht der Transfer-Typ bei vielen FTP-Servern nach der Etablierung einer neuen Verbindung aber auf A.

Datenstruktur

Die Datenstruktur ist ein weiteres Merkmal, das vor einer Übertragung eingestellt werden muss. Die Optionen sind:

- F. Datei (File)
- R. Datensatz (Record)
- P. Seite (Page)

R und P sind nur selten implementiert, beispielsweise bei FTP-fähigen Datenbanken. Die Einstellung erfolgt mit

STRU F

Weiter verbreitet ist dagegen die Angabe des Transfer-Modus mit dem Kommando MODE. Auch hier sind drei Optionen möglich:

Transfer-Modus

- S. Stream-Mode für kontinuierliche Übertragung.
- B. Block-Mode für die Zerlegung in Blöcke mit eigenen Headern.
- C. Compress-Mode für die Komprimierung von Daten (RLE).

Die Standardeinstellung lautet:

MODE S

Normalerweise liegt die Kontrolle des Verbindungsaufbaus beim Server. Wenn ein FTP-Client eine Verbindung aufbaut, werden nur die IP-Adresse und Port-Nummer übertragen. Der FTP-Server speichert diese Werte, beendet die anfordernde Verbindung und baut dann seinerseits eine neue auf. Das funktioniert, solange der Weg zwischen Server und Client in beiden Richtungen frei ist. Oft sitzen die Clients jedoch hinter einem Gateway oder einer Firewall. Dann erreicht der Server den Client mit der übergebenen Adresse nicht mehr. Um dieses Problem zu umgehen, gibt es den passiven Modus. Mit dem Kommando PASV teilt der Client mit, dass der Server passiv kommunizieren soll. Der Server sendet nun seinerseits IP-Adresse und Portnummer für die Kommunikation und der Client baut die Verbindung in der gewünschten Form auf.

Passiver Modus

FTP-Statuscodes

Auch FTP verwendet einen Statuscode zur Beantwortung von Anfragen. Wie bei HTTP und SMTP genügt es oft, nur die erste Ziffer auszuwerten, um Fehlerzustände oder normal verlaufende Operationen zu erkennen:

- 1. Neutrale Antwort, unaufgeforderte Meldung.
- 2. Positive Antwort, Kommando erfolgreich verarbeitet.
- 3. Positive Antwort mit der Bitte um weitere Informationen.
- 4. Fehler, das Kommando kann zeitweilig nicht beantwortet werden, Wiederholung möglich.
- 5. Fehler, Wiederholung zwecklos, Kommando falsch oder Server nicht verfügbar.

Die mit 1xx beginnenden Statuscodes gibt es nur bei FTP. Sie sind besonders schwierig zu verarbeiten, denn die Absendung durch den

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Server kann zu jeder Zeit erfolgen, also auch während der Datenübertragung oder zwischen anderen Kommandos und Meldungen. Sie ersetzen jedoch nicht die normalen Antworten. Jedes Kommando wird garantiert mit mindestens einem 2xx – 5xx-Kommando beantwortet. Folgende Kommandos können von 1xx-Statuscodes begleitet werden:

- APPE, LIST, NLST, REIN, RETR, STOR, STOU

Ablauf der Kommunikation

Der folgende Abschnitt zeigt den Ablauf einer typischen Kommunikation zwischen Client und Server mit dem Protokoll FTP:

*Listing 2.2:
Typischer Ablauf
einer FTP-Verbin-
dung*

```
Client: (Verbindungsaufbau mit FTP-Client)
Server: 220-Service ready
Server: (optional Informationen zur Authentifizierung)
Server: 220-Service ready
Client: USER anonymous
Server: 331 guest login ok, send e-mail as password
Client: PASS joerg@krause.net
Server: 250 guest login ok, access restrictions apply
Client: CWD ftp/download/
Server: 250 CWD command succesfull
Client: PWD
Server: 257 "ftp/download/" is current directory
Client: TYPE I
Server: 200 TYPE set to I
Client: PASV
Server: 227 Entering Passive Mode (62,208,3,4,4,23)
Client: RETR servicepack5.exe
Server: 150 Opening Data Connection
Server: (sendet Daten)
Server: 226 Transfer complete
Client: QUIT
Server: 221 Goodbye
```

Das Beispiel zeigt eine Authentifizierung als anonymer Nutzer, einen Verzeichniswechsel und einen Download einer Datei. Zur Übertragung (im Binärformat) wird außerdem der passive Modus verwendet.

IP-Adresse erkennen

Das PORT-Kommando und die Antwort auf PASV enthalten die zu verwendende IP-Adresse und den Datenport. Wie in Listing 2.2 zu sehen war, erfolgt die Angabe als kommaseparierte Liste. Das Format der Liste hat folgenden Aufbau:

```
IP1, IP2, IP3, IP4, PORT1, PORT2
```

Sie kennen den Aufbau einer IP-Adresse nach folgendem Schema:

```
IP1.IP2.IP3.IP4:PORT1,PORT2
```

Jede Zahl umfasst ein Byte. Da Portnummern 16-Bit breit sind, müssen für die Angabe zwei Byte angegeben werden. Die Adresse 1024 würde also als 4,0 geschrieben werden. Zur Umrechnung multiplizieren Sie einfach das höherwertige Byte mit 256.

Im Internet herrscht ein zunehmender Mangel an IP-Adressen (siehe auch ➡ Abschnitt *IP-Adressversionen* ab Seite 63). Deshalb wurde bereits vor einigen Jahren ein neues Adresssystem entworfen. Offensichtlich ist aber ein Teil des Mangels politisch bedingt und so konnte sich IPv6 nicht so schnell wie erhofft durchsetzen. Dennoch sind die Protokolle auf die Umstellung vorbereitet. Da FTP unmittelbar mit IP-Adressen umgeht, ist eine Erweiterung erforderlich. Neu sind die Kommandos LPTR (Long Port) und LPSV (Long Passive). In der RFC 1639 ist die Syntax beschrieben.

Umgang mit IPv6-Adressen

RFC 1639

FTP wird häufig eingesetzt, um große Datenmengen zu übertragen. Dabei kann es leicht zu Leitungsstörungen kommen. Bei direkten Verbindungen zwischen FTP-Servern oder beim Einsatz von ISDN ist die Störanfälligkeit verhältnismäßig gering. Häufiger werden jedoch Nutzer per Modem auf Server zugreifen. Wenn eine 1 MByte große Datei nach 980 000 Byte abreißt, ist dies ausgesprochen ärgerlich. Das Standardverfahren der Datenübertragung, Stream, ist also nur bedingt geeignet. Es ist allerdings auch die schnellste Form der Übertragung.

Wiederaufnahme der Übertragung

Damit stellen Sie das Block-Verfahren ein:

Block-Verfahren

MODE B

Dabei zerlegt der Server die Datei in Blöcke, versieht jeden Block mit einem eigenen Header und sendet sie einzeln an den Client. Reißt die Verbindung ab, kann der Client die schon empfangenen Blöcke speichern und die nach der erneuten Verbindungsaufnahme eintreffenden Blöcke richtig zuordnen. Allerdings unterstützen nicht alle FTP-Server die erweiterten Transfermodi B (Block) und C (Compressed). Die Anforderung der übrigen Blöcke erfolgt mit dem Kommando REST.

Probleme mit FTP

FTP ist ein sehr altes Protokoll. Die Ausgaben der Kommandos LIST und NLST sind nicht ausreichend standardisiert. Eine Angabe zur Übertragung der Dateilänge gibt es nicht. Intelligente Clients speichern die Angaben des LIST-Kommandos und geben den Wert dann bei einem nachfolgenden GET an. Ob die Datei tatsächlich im Augenblick der Übertragung diese Größe hat, wissen sie nicht. Dateien und Verzeichnisse können kaum unterschieden werden. Praktisch bleibt der Versuch, sichere Angaben über die Größe der nächsten zu ladenden Datei zu machen, ein Wagnis.

Unklar: Angabe der Dateigröße bei der Übertragung

Alternativen zu FTP gibt es derzeit nicht. Die Nachteile werden zwar von anderen Entwicklungen vermieden, ausreichende Verbreitung fand indes keines der möglicherweise zu diesem Zweck einsetzbaren Protokolle wie LDAP, NDS oder WebNFS.

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

2.2.3 Protokolle für E-Mail-Verkehr

Das Versenden von E-Mails mit einem SMTP-Server ist auf allen Betriebssystemen Standard. Aus diesem Grund lohnt es sich, einen Blick auf das dabei hauptsächlich genutzte höhere Protokoll zu werfen: das *Simple Mail Transfer Protocol*. Dieses dient dem Austausch von Mailnachrichten zwischen Clients, die SMTP verwenden, und einem Server, der SMTP zum Empfang und zur Weiterleitung einsetzt.

Des Weiteren gibt es noch wichtige Protokolle, die vor allem clientseitig die Art und Weise des Mailempfangs steuern: POP3 und IMAP4. POP3 wird in PHP über die Netzwerkfunktionen unterstützt, IMAP durch eine eigene Funktionsklasse.

SMTP

RFC 821 und 2821	Der SMTP-Dienst basiert auf RFC 821 und unterstützt darüber hinaus bestimmte Erweiterungen, die unter dem Begriff <i>Extended SMTP</i> (ESMTP) bekannt sind. Heute aktuell ist RFC 2821 (April 2001).
Abwärtskompatibilität sichergestellt	ESMTP umfasst eine einheitliche Beschreibung für Erweiterungen von Mailservern. Dabei kann jeder Hersteller eigene Erweiterungen in seine SMTP-Serverimplementierung einbringen. Ein ESMTP-Server und ein ESMTP-Client können einander erkennen und sich über die verfügbaren erweiterten Funktionen austauschen. Beide, Server wie auch Client, müssen allerdings Abwärtskompatibilität sicherstellen und die grundlegenden SMTP-Befehle (gemäß RFC 821 bzw. neuerdings 2821) beherrschen.
SMTP-Client und -Server	Bei der Beschreibung der SMTP-Funktionen werden im weiteren Text die Begriffe Client und Server benutzt. Genau genommen handelt es sich eigentlich um SMTP-Sender und SMTP-Empfänger, da ein SMTP-Server Mails an einen anderen SMTP-Server senden (weiterleiten) kann. Damit ist dieser dann wiederum der »Client« bzw. Sender und der andere Server der »Server« bzw. Empfänger.
UA und MTA	Der Benutzer bzw. die Mailanwendung kommuniziert direkt nur mit dem <i>User Agent</i> (UA) genannten Teil des SMTP-Dienstes. Dieser sorgt für die Übernahme der Maildaten und die Weitergabe an den eigentlichen Sendeprozess. Die SMTP-Sender- und Empfängerprozesse werden auch als <i>Message Transfer Agents</i> (MTA) bezeichnet.
Die SMTP-Kommandos	In der folgenden Tabelle finden Sie den minimalen SMTP-Befehlssatz, der durch jeden SMTP-Client und -Server unterstützt wird.

Tabelle 2.15:
Minimaler SMTP-
Befehlssatz

Befehl	Beschreibung
HELO <sender>	Eröffnet die Verbindung von einem SMTP-Client (dem Sender) zum SMTP-Server.

Befehl	Beschreibung
EHLO <sender>	Eröffnet wie HELO die Verbindung von einem ESMTP-Client (dem Sender) zum ESMTP-Server. Dieses Kommando ist deshalb mit in dieser Tabelle aufgenommen worden, da es heute die Standard-Eröffnungsprozedur zwischen Client und Server darstellt.
MAIL FROM: <maddr>	Beginn einer Mail. Geben Sie hier die Absender-E-Mail-Adresse an.
RCPT TO: <maddr>	Empfänger der Mail. Sie können auch mehrere Empfänger festlegen, indem Sie den Befehl wiederholen.
DATA	Initiiert die Eingabe des Nachrichtentextes; Alles, was Sie jetzt zeilenweise eingeben, wird als Mailtext aufgenommen. Schließen Sie die Eingabe mit einem Punkt ».« ab, der allein am Beginn einer Zeile stehen muss (genau genommen zwischen zwei CRLF).
QUIT	Beendet die Verbindung zum SMTP-Server.
RSET	Reset. Beendet die Verbindung und die laufende Mailtransaktion.

Der SMTP-Server meldet den Erfolg oder Fehlschlag von Operationen mit dreistelligen Codes. Dabei hat jede Stelle eine bestimmte Bedeutung. Die Bedeutung der ersten Ziffer finden Sie in der folgenden Tabelle:

SMTP-Rückmeldungen

Code	Beschreibung
1xx	Positive vorbereitende Antwort Diese Codes können nur durch ESMTP-Server zurückgegeben werden und zeigen an, dass das zuvor gesendete Kommando zwar akzeptiert worden ist, allerdings weitere Anweisungen benötigt werden, um die Aktion abzuschließen.
2xx	Positive komplette Antwort Die geforderte Aktion konnte erfolgreich abgeschlossen werden und es kann ein neues Kommando durch den Client ausgelöst werden.
3xx	Positive Zwischenantwort Das Kommando wurde akzeptiert. Der Server wartet auf weitere Daten, wie beispielsweise beim DATA-Befehl. Hier werden so lange Eingabedaten (Textzeilen) angenommen, bis sie durch das Zeichen ».« abgeschlossen werden (siehe Tabelle 2.15).

Tabelle 2.16:
SMTP-Server-
Antwortcodes:
1. Ziffer

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Code	Beschreibung
4xx	Vorläufig negative Antwort Das Kommando ist nicht akzeptiert und damit die angeforderte Aktion nicht durchgeführt worden. Der Fehlerstatus ist allerdings nur temporär. Dies bedeutet, dass zwar die Aktion zunächst fehlgeschlagen ist, aber eine Wiederholung des Kommandos durchaus noch zum Erfolg führen kann.
5xx	Permanente negative Antwort Das Kommando wurde auch hier nicht akzeptiert, allerdings ist der Fehlerstatus permanent. Eine Wiederholung des Kommandos mit den gleichen Einstellungen führt garantiert wieder zu diesem Fehlercode.

Die folgende Tabelle enthält die Bedeutung der zweiten Ziffer des dreistelligen SMTP-Codes:

Tabelle 2.17:
SMTP-Server-
Antwortcodes:
2. Ziffer

Code	Beschreibung
x0x	Syntaxfehler Kennzeichnet Fehlermeldungen, die aufgrund von Syntaxfehlern (falsches oder nicht unterstütztes Kommando etc.) verursacht worden sind.
x1x	Informationen Kennzeichnet Antworten, die Informationen zurückgeben (beispielsweise Statusmeldungen).
x2x	Verbindung Kennzeichnet Meldungen, die im Zusammenhang mit dem Verbindungs- bzw. Übertragungsstatus stehen.
x5x	Mailsystem Codes in Bezug zu Meldungen des Mailsystems

Die dritte Ziffer ermöglicht eine etwas feinere Abstimmung für Meldungen, die durch die zweite Ziffer bestimmt werden. Auf diese wird in diesem Rahmen nicht näher eingegangen. Abschließend zu diesem Thema finden Sie in der folgenden Tabelle typische SMTP-Codes mit ihren Bedeutungen.

Tabelle 2.18:
SMTP-Codes mit
ihren Bedeutungen

Code	Beschreibung
211	Systemstatus oder System-Hilfemeldungen
214	Hilfemeldungen Damit werden Meldungen gekennzeichnet, die direkte Hilfestellung zum System bzw. zu einzelnen, nicht standardisierten Befehlen geben.
220	<domainname> Service bereit

Code	Beschreibung
221	<domainname> Service schließt den Verbindungskanal
250	Angeforderte Mailaktion OK, vollständig abgeschlossen
354	Beginn der Mail-Eingabe; Ende mit ».«
421	<domainname> Service nicht verfügbar; Verbindungskanal wird geschlossen
450	Angeforderte Mailaktion nicht durchgeführt, da Mailbox nicht verfügbar (beispielsweise im Falle einer Überlastung)
451	Angeforderte Aktion abgebrochen, da ein lokaler Fehler in der Abarbeitung aufgetreten ist
452	Angeforderte Aktion nicht durchgeführt, zu knapper System-speicher
500	Syntaxfehler bzw. Kommando nicht erkannt (kann auch eine zu lange Kommandozeile auslösen)
501	Syntaxfehler in übergebenen Parametern/Optionen
502	Kommando nicht implementiert
503	Falsche Reihenfolge von Kommandos
504	Kommando-Parameter nicht implementiert
550	Angeforderte Mailaktion nicht durchgeführt, da Mailbox nicht verfügbar (beispielsweise Mailbox nicht existent oder Sicherheitsrichtlinien lassen keinen Zugriff zu)
552	Angeforderte Aktion nicht durchgeführt, da kein Systemspeicher mehr verfügbar ist
553	Angeforderte Aktion nicht durchgeführt, da der Name der Mailbox ungültig ist (beispielsweise Syntaxfehler im Name)
554	Übertragung fehlgeschlagen

POP3

POP3 dient zum Abholen von E-Mails von einem Mail-Server. Ein POP3-Server empfängt E-Mails und speichert sie so lange, bis der Nutzer sie abholt. RFC 1939 definiert POP3. In der RFC 1957 finden Sie einen Bericht zur aktuellen Lage der Nutzung von POP3-Servern im Internet.

POP3 als Protokoll wird vom *mailfrom*-Dienst genutzt, der standardmäßig auf Port 110 auf Kommandos lauscht. Tabelle 2.19 zeigt die wichtigsten Kommandos, die für eine reibungslose Kommunikation benötigt werden.

POP3
RFC 1939
RFC 1957

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Tabelle 2.19:
POP3-Kommandos
und erweiterte
Kommandos
(Auszug)

Kommando	Parameter	Beschreibung
<i>Standardkommandos</i>		
DELE	Nachrichtennr.	Markiert die angegebene Mail zum Löschen nach Ausführung von QUIT.
LIST	[Nachrichtennr.]	Listet alle Mails mit Nummer und Größe auf. Wird die Nummer angegeben, wird die Größe der betreffenden Mail gezeigt.
NOOP		Verbindungsüberprüfung
QUIT		Verbindung beenden
RETR	Nachrichtennr.	E-Mail anfordern
RSET		Löschmarkieren aufheben
STAT		Anzahl und Größe aller Mails abfragen
USER	Nutzername	Nutzer anmelden
PASS	Kennwort	Nutzer anmelden (nach USER)
<i>Erweiterte Kommandos</i>		
APOP	Name Kennwort	Anmeldung, Kennwort wird verschlüsselt
TOP	Nachrichtennr. Anzahl	Holt eine Anzahl Zeichen aus der Mail <i>Nachrichtennr.</i>
UIDL	[Nachrichtennr.]	ID einer oder aller E-Mails

Der POP3-Server vergibt für jede im Postfach liegende Nachricht eine fortlaufende Nummer. Nachdem das Postfach geleert wurde, beginnt die Nummerierung wieder mit 1. Der Aufbau einer Verbindung wird sofort mit der Statusmeldung des Servers beantwortet, ein eigenes Startkommando ist nicht notwendig. Sind am POP3-Server Name und Kennwort erforderlich, folgen nun die Kommandos USER und PASS. Beachten Sie, dass der normale POP3-Server ohne erweiterte Kommandos auch das Kennwort unverschlüsselt überträgt.

Statuscodes

Antwortzeilen enthalten einen einfachen Statuscode:

- +OK Meldungstext

Das Kommando wurde verstanden und konnte ausgeführt werden. Der Meldungstext enthält die erwarteten Daten.

- -ERR Fehlertext

Das Kommando wurde nicht erkannt oder konnte nicht ausgeführt werden. Der Fehlertext erklärt, welches Problem vorlag.

Die im Gegensatz zu SMTP fehlenden Statuscodes machen die Weiterverarbeitung von Fehlermeldung nicht besonders einfach. Auch moderne Mailclients wie Outlook geben deshalb die Fehlermeldung unverändert an den Nutzer weiter. Sie können natürlich auf den Fehler schließen, indem Sie das zuvor gesendete Kommando betrachten.

Ein POP3-Server gibt immer eine Zeile zurück. Einige Kommandos geben mehrere Zeilen (beispielsweise die E-Mail-Daten) zurück. Dann endet das Kommando mit der Folge »Leere Zeile-Punkt-Leere Zeile«.

2.2.4 Grundlagen der E-Mail-Programmierung

Der erste Teil befasst sich mit der Kodierung von E-Mail-Nachrichten. Am Ende werden die in der RFC 822 definierten Header der E-Mail-Nachrichten diskutiert.

Kodierung und Dekodierung

Internet E-Mail und Nachrichten für das Usenet wurden entwickelt, um Texte zu übermitteln. Viele Systeme verhalten sich auch heute noch so, als ob nur ASCII-Zeichen übermittelt werden. Der ASCII-Zeichensatz ist lediglich 127 Zeichen groß und nutzt das Bit 8 nicht, dieses Bit ist immer 0. Entsprechend spricht man auch von 7-Bit-Zeichen. Bei der Übermittlung von Sonderzeichen, Umlauten oder auch binären Daten stößt man mit solchen Systemen auf Schwierigkeiten, denn hier werden alle 8 Bit verwendet.

Für die Verarbeitung von Anhängen für E-Mail muss also eine Form der Kodierung gefunden werden, die die Binärdaten (8-Bit) in das 7-Bit-Format überführt. Im Laufe der Zeit haben sich gleich mehrere Standards dazu herausgebildet:

- UUencode
- MIME
- Base64
- Quoted-Printable
- Binhex

Die Vielfalt macht die Vorteile des Standards wieder zunichte. Außer der reinen Dekodierung ist es auch eine Herausforderung, den verwendeten Standard überhaupt zu erkennen. Die einzelnen Standards



werden im Folgenden detailliert vorgestellt. Dies ist die Grundlage für die unproblematische Nutzung entsprechender Funktionen in PHP.

UUencode

UUencode

Der Name UUencode steht für Unix-to-Unix-Encode. Entsprechend verbreitet ist dieser Standard in der Unix-Welt. Er wurde ursprünglich entwickelt, um Dateien von einem Unix-Computer zu einem anderen zu transportieren. UUencode hat eine weite Verbreitung gefunden, da in den Anfangsjahren des Internets überwiegend Unix-Computer für den E-Mail-Austausch zum Einsatz kamen (und auch heute noch diesen Teil dominieren).

Die meisten E-Mail-Programme können UUencode-Dateien dekodieren. Kodieren gehört dagegen meist nicht zu den angebotenen Leistungen. Darüber hinaus wird UUencode im Usenet stärker verwendet als bei E-Mail. Die nahe Verwandtschaft beider Dienste lässt es aber sinnvoll erscheinen, diese hier gemeinsam zu behandeln.

UUencode-kodierte Dateien sind etwa 42% größer als das Original. Das liegt zum einen am Verzicht auf 1 Bit, zum anderen an den zusätzlich untergebrachten Informationsdaten.

UUencode-Daten haben einen festen Aufbau:

```
begin 644 a.gif
M1TE&.#EA)0`H`+,``.P`2QBQ`/____M
RP`~~~~)0`H`~~$5S#(2:N]..0-N_]@*(
MB@ (LH%ZM^U;Q3,6L+>6MW@>_5VXW%,J`Q500>5PUF:HE\4F20ITX'# :K-5FG
;WB3MZR&#J^(QM9RVF'7PN'Q.K)00^+Q^!``[
` end
```

Die erste Zeile enthält das Wort »begin«, gefolgt von einem Leerzeichen und einer dreistelligen Zahl (644). Diese entspricht den Unix-Dateirechten (read/write/execute). Andere Betriebssysteme ignorieren diese Angaben. Dann folgt der Name der Datei (a.gif). Der eigentliche Inhalt folgt auf den nächsten Zeilen. Jede Zeile beginnt mit der Zeilenlänge, das Zeichen M steht dabei für 60, es folgen also 60 Zeichen im 7-Bit-Format. Die letzte Zeile ist normalerweise kürzer, damit die tatsächliche Dateilänge abgebildet wird. Der Datenblock wird mit einem Apostroph und dem Wort »end« abgeschlossen.

MIME

MIME

MIME steht für *Multipurpose Internet Mail Extensions*. Dieses Format wurde nicht für Mail-Anhänge entwickelt, sondern für die Übertragung strukturierter (mehrteiliger) Nachrichten. Eine Nachricht im MIME-Format muss nicht notwendigerweise Anhänge enthalten. Häufig wird für die Anhänge die Kodierung nach Base64 genutzt oder Quoted-Printable (siehe weiter unten).

Die offizielle Beschreibung des MIME-Standards finden Sie in der RFC 2045. Dies ist vor allem im Hinblick auf die Erweiterung der RFC 822 zu sehen, die den Mailstandard an sich beschreibt, jedoch grundsätzlich nur Textnachrichten vorsieht.

RFC 2045

Neuere Mailprogramme unterstützen MIME problemlos. Bei sehr alten Programmen kann es Probleme geben. Eine MIME-Dekodierung kann also eine sinnvolle Aufgabe sein. Manche Programme bieten beim Versenden von E-Mail oder News die Auswahl zwischen MIME und UUencode. Dies ist irreführend, da MIME selbst nicht kodiert. Wenn UUencode ausgewählt wird, überträgt das Programm die Nachricht im Textformat und kodiert Anhänge mit UUencode. Wenn MIME ausgewählt wird, entsteht eine Nachricht mit MIME-Headern und mit Base64- oder Quoted Printable-kodierten Anhängen.

Der wesentliche Unterschied ist also die Existenz eines MIME-Headers, einer einleitenden Beschreibung des folgenden Textes. Ein typischer Header sieht so aus:

```
MIME-Version: 1.0
Content-Description: "Base64 encode of a.gif by Codec"
Content-Type: image/gif; name="a.gif"
Content-Transfer-Encoding: Base64
Content-Disposition: attachment; filename="a.gif"
```

Entscheidend ist die in der Zeile Content-Type angegebene Information zur Art des Inhalts. Zulässige Content-Typen sind in der RFC 2046 beschrieben. Diese werden auch im Zusammenhang mit der Webserver-Programmierung auftreten. Allerdings folgen den MIME-Headern im Browser keine kodierten Daten, da der Browser direkt mit Binärdateien umgehen kann. Das im Beispiel angehängte Bild »a.gif« würde ohne Umwege zum Browser gelangen. Die eigentliche Kodierung mit Base64 oder Quoted Printable wird nachfolgend beschrieben.

Beim Umgang mit MIME sollten Sie die Haupt- und Subtypen kennen, die MIME definiert. Eine vollständige Liste ist in RFC 2046 zu finden. Tabelle 2.20 zeigt die gängigsten Typen.

**MIME-Typen
RFC 2046**

Haupttyp	Subtyp	Beschreibung
application	octet-stream	Binäre Daten ohne Verwendungsangabe
application	postscript	Postscript-Datei
application	pdf	PDF-Datei (Adobe)
application	word	Microsoft Word-Datei
audio	basic	Audiodaten
image	jpeg	JPG-Bild
message	rfc822	E-Mail nach RFC 822 mit Header

Tabelle 2.20:
MIME-Media-Typen
(Auswahl)

Haupttyp	Subtyp	Beschreibung
model	vrml	VRML-3D-Objekt
multipart	mixed	Mehrteilige Nachricht
text	plain	ASCII-Text
text	html	HTML-Seite
video	mpeg	MPEG-Video

Die Media-Typen sollen dafür sorgen, dass der Empfänger der Nachricht die passende Applikation zur Anzeige oder Ausführung der Dateien bereitstellt. Entsprechend umfangreich ist die Zahl der verfügbaren Subtypen. Auf die tatsächliche Ausführung kann man sich übrigens nicht verlassen. Vor allem neuere Erweiterungen aus der PC-Welt, wie *word* oder *rtf*, finden auf Unix-Systemen keine Entsprechung.

text/*

Der häufigste Grundtyp ist *text*. Normalerweise folgt als Subtyp */plain*:

```
Content-type: text/plain; charset=us-ascii
```

Alternativ werden auch HTML-Dateien mit dem Text-Typ verschickt:

```
Content-type: text/html
```

Der Zeichensatz kann optional mit angegeben werden, dazu wird die Zeile um den Parameter *charset* erweitert.

application/*

Alles, was nicht Text ist und nicht sonstigen Typen zugeordnet werden kann, wird als *application* gekennzeichnet. Dahinter sollte sich ein Programm verbergen, das die übertragene Datei anzeigt oder auf andere geeignete Weise verarbeitet. Auch Verschlüsselungsprogramme werden über diesen Typ angesprochen. Ohne weitere Angabe wird der Subtyp *octet-stream* verwendet. Das Clientprogramm wird dann die Datei zum Speichern anbieten und keiner Applikation direkt zuordnen. In der Windows-Welt sind die Subtypen *rtf* und *word* verbreitet, in der Unix-Welt eher *postscript*.

Benutzerdefinierte Subtypen sind möglich, diese sollten mit dem Präfix »x-« gekennzeichnet werden. So wird eine Excel-Datei mit

```
Content-type: application/x-excel; name=charts.xls
```

übertragen. Ob der Empfänger damit etwas anfangen kann oder will, spielt für die Angabe keine Rolle.

image/*

Bilder, die direkt in der Nachricht eingebettet sind, werden mit dem Haupttyp *image* gekennzeichnet. Dabei wird davon ausgegangen, dass das Clientprogramm in der Lage ist, Bilddaten zu erkennen und darzustellen. Als Subtypen kommen *jpeg*, *gif* und *png* in Frage. Seltener trifft man auch auf *g3fax* (Gruppe 3-Faxe).

Neben dem Standardsubtyp *rfc822* können auch *partial* und *external-body* angegeben werden. Die Übertragung der Nachricht muss in diesen Formaten unkodiert erfolgen. Nachrichten nach *rfc822* bestehen aus einem RFC 822-konformen Header und Body und eignen sich zum Weiterleiten von Nachrichten. Die Hauptanwendung sind Standard-E-Mails.

Mit *partial* kann eine Nachricht in mehrere Teile zerlegt werden. Aufteilen und Zusammensetzen ist Sache der Clientapplikationen. Regeln zum Aufteilen sind in RFC 2046 definiert.

Beschreibende Informationen werden mit *external-body* definiert. Mit den zusätzlichen Attributen *name*, *site* und *access-type* werden Informationen über die zu nutzende Ressource angegeben.

Außerhalb des MIME-Standard wurde in der RFC 2077 der Haupttyp *model* definiert. Hintergrund war die Mitte der 90er Jahre aufkommende 3D-Welle im Internet. So werden als Subtyp *vrml*, *iges* und *mesh* definiert. Einzig VRML (Virtual Reality Model Language) konnte sich etwas verbreiten. Inzwischen gibt es kaum noch ernsthafte Anwendungen, VRML kann problemlos durch Java ersetzt werden.

Zum Aufteilen einer Nachricht kann prinzipiell *message/partial* genutzt werden. Nur sind dann alle Subtypen identisch. Soll eine Nachricht dagegen aus mehreren Subtypen bestehen, bietet sich der Haupttyp *multipart* an. Zulässige Subtypen sind *mixed* (verschiedene Teile), *parallel*, *digest* und *alternativ*. Multipart-Nachrichten sind besonders häufig anzutreffen und sollten bei eigenen Projekten unbedingt beachtet werden. Der Aufbau wird durch den Content-Type gesteuert. Der Empfänger weiß nun, dass der nächste Abschnitt Teil einer Nachricht ist und dass am Ende ein Trennzeichen folgt. Die Nachricht beginnt nach zwei aufeinander folgenden Zeilenwechselln. Das Ende besteht aus einem Zeilenwechsel, zwei Minus-Zeichen und der angegebenen Trennzeichenfolge. Der komplette Aufbau besteht aus:

1. Transport-Header
2. Zwei Zeilenwechsel
3. Vorspann
4. Trenner
5. Nachricht
6. Trenner
7. Header der zweiten Nachricht
8. weiter bei Punkt 2

Die folgende Musternachricht zeigt die Anwendung, die Symbole <CRLF> kennzeichnen Zeilenwechsel, die unbedingt an diesen Stellen stehen müssen.

*Listing 2.3:
Aufbau einer
einfachen Multipart-
Nachricht*

```
From: joerg@krause.net
To: lektor@verlag.de
...
Subject: Buch ist fast fertig!<CRLF>
MIME-Version: 1.0<CRLF>
Content-type: multipart/mixed; boundary="trenner"<CRLF>
<CRLF>
Im Vorspann steht der Klartext fuer Nutzer, deren Client
MIME/Multipart nicht versteht. <CRLF>
--trenner<CRLF>
Hier folgt der erste Teil der Nachricht, dieser Teil hat im
Beispiel keinen eigenen Header<CRLF>
--trenner<CRLF>
Content-type: text/html; charset=iso8859-1<CRLF>
Content-transfer-encoding: Quoted-Printable<CRLF>
<CRLF>
Hier folgt ein Teil mit Header; es handelt sich um eine HTML-
Datei<CRLF>
--trenner<CRLF>
Was hier steht, wird ignoriert<CRLF>
```

Beachten Sie, dass die Zeichenkette für die Trennung nicht mehr als 70 Zeichen lang sein darf. Es hat sich eingebürgert, Folgen von Minus- oder Gleichheitszeichen zu nutzen. Neben Buchstaben und Zahlen sind die folgenden Zeichen zulässig:

' () + _ - . / : = ?

Die Applikation, mit der die Nachricht erstellt wird, muss dafür sorgen, dass die Trennungszeichenkette eindeutig ist und nicht als Teil der Nachricht auftritt. Dies ist sehr einfach sicherzustellen. Der Body der Nachricht kann normalerweise nur nach Quoted Printable oder Base64 kodiert werden. Darin kann die Zeichenfolge =_ nicht vorkommen. Ein Trennzeichen der Art

=_Multipart Separator_=

genügt also völlig diesem Anspruch und ist überdies für Clients gut lesbar, die nicht selbst die Trennung der Teile besorgen können.

Geschachtelte Nachrichten

Multipart-Nachrichten definieren den Content-type der folgenden Nachricht immer wieder neu. Es spricht also nichts dagegen, auch hier wieder eine Multipart-Nachricht anzuschließen. Tatsächlich sind solche rekursiven Nachrichtenkonstrukte erlaubt. Um die Nachricht aus dem Chaos wieder korrekt zusammensetzen zu können, werden die folgenden Subtypen verwendet:

- alternativ

Die Teilnachrichten enthalten die gleiche Information in verschiedenen Formaten. Dies wird häufig eingesetzt, um eine E-Mail als Plain-Text und zugleich als HTML-Datei zu versenden. Der Client kann dann entscheiden, welche Form er zur Anzeige bringt.

- digest

Jeder Teil ist nach RFC 822 (als message/rfc822) formatiert. Wird oft in Mailinglisten eingesetzt, wo viele E-Mails zu einer Nachricht (dem Digest) zusammengesetzt werden. Nur wenige Clients können allerdings diese Konstruktion wieder in einzelne E-Mails zerlegen.

- mixed

Inhalt ist ohne nähere Spezifikation.

- parallel

Der Teil enthält Daten, die Bestandteil einer anderen Nachricht sind. Der Client sollte aus allen Teilen die Gesamtnachricht zusammensetzen.

Verschachtelte Nachrichten sollten auch bei der Angabe der Trennungszeichenkette so gestaltet werden, dass Clients, die Multipart nicht verarbeiten können, lesbare Texte ausgeben. Üblich sind verkürzte Zeichenketten:

```
=_MPM_
```

Base64

Base64 wird als Standard für die Übermittlung von Anhängen in MIME-E-Mails verwendet. Es gibt zwar keinen Zwang, beide Methoden miteinander zu verbinden, reine Base64-Mails sind aber extrem selten zu finden. **Base64**

Base64 vergrößert eine Datei um etwa 37%. Wenn es um die Effizienz des Verfahrens geht, ist Base64 also besser als UUencode. Eine typische Kodierung hat folgendes Aussehen:

```
MIME-Version: 1.0
Content-Description: "Base64 encode of a.gif by Codec"
Content-Type: image/gif; name="a.gif"
Content-Transfer-Encoding: Base64
Content-Disposition: attachment; filename="a.gif"
R0lGODlhJQAOALMAAQwASxixAP////////////////////////////////
////////////////yWAAAAAJQAOAAAEVzDISau900vNu/9gKI5kaZ5oigIsoF6t+1bxT
MwSLewt3ge/V243FMqAVQqEVw1mao18UmSQp04HDarNVmn3iTt6yGDq+IxtZy2mHXw
uHx0r9vv+Lx+BAA7
```


Quoted Printable

Quoted Printable

Dieses Verfahren wird ebenso wie Base64 mit MIME angewendet. Es wird von vielen E-Mail-Programmen automatisch genutzt, wenn der Dateianhang überwiegend ASCII-Zeichen und nur wenige zu kodierende Zeichen enthält. Dies ist vor allem bei nichtenglischen Texten der Fall. Druckbare Zeichen werden nicht umgewandelt – nur die Sonderzeichen aus dem erweiterten ASCII-Satz. Dabei wird das Gleichheitszeichen als Trennsymbol eingesetzt, gefolgt von dem Hexadezimalen Code des Zeichens. Das Gleichheitszeichen selbst sieht dann so aus: =3D.

Quoted Printable vergrößert normalen Text lediglich um 3%. Im Extremfall, wenn nur Sonderzeichen verarbeitet werden, können es aber auch 200% sein. Der Einsatz ist weniger der Anhang einer E-Mail als mehr der Text der E-Mail selbst. Quoted Printable kann auch das ursprüngliche Layout einer Nachricht bewahren. Dazu werden vollständige Absatzumbrüche eingefügt. Wenn ein Client den automatischen Zeilenumbruch nicht unterstützt, werden Absätze als lange Zeilen angezeigt. Quoted Printable kodiert weiche Zeilenumbrüche mit einem alleinstehenden Gleichheitszeichen am Zeilenende. Diese Umbrüche sollten nicht ausgeführt werden, wenn die Zeilenbreite des Anzeigeprogramms größer ist als die ursprüngliche Zeilenlänge in der Nachricht. Führende Leerzeichen in einer Zeile werden entfernt. Sind sie dennoch erwünscht, kodiert man sie mit =20 am Ende der Zeile. Quoted Printable-Zeichen werden an der Kombination des Gleichheitszeichens mit zwei unmittelbar folgenden Hexadezimalziffern erkannt. Das folgende Beispiel zeigt einen kodierten Text, wie er angezeigt wird, wenn keine korrekte Kodierung erfolgt.

```
MIME-Version: 1.0
Content-Type: text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: quoted-printable
This is a example of a quoted-printable text file. =
Some special characters used: =20 equal sign =3D, =
dollar sign $, or even extended characters such as =
the cent sign =A2 or foreign characters =C0=C6=DF
```

Fehlt die Kodierung, bleibt der Text normalerweise lesbar.

BinHex

Diese Form der Kodierung kommt aus der Macintosh-Welt und ist dort weit verbreitet. Kodierer und Dekodierer existieren aber inzwischen auch für alle anderen Plattformen. Der wesentliche Unterschied besteht in der Art der Kodierung: BinHex komprimiert die Daten zusätzlich. Solche Dateien können dann kleiner sein als das Original. Wenn allerdings bereits komprimierte Daten kodiert werden, ist das Ergebnis ca. 40% größer. Die Kodierung hat folgendes Aussehen:

```
(This file must be converted with BinHex 4.0)
: "%ZCfPQ!&4&@&4YC'pc!*!&SJ#3"*!!bNG*4MJjB58!+!#c!!$X!%XBx3$rN#S!*
!*3!S!!%9c$)5DZp11[0ZrpJ+)jNDCjSLJ)XS&kYqe
Ea6-@X,H0YhJHr9fi&!a933H9`eQDSPm8Q53Tdi($DV09QRhL6Ykb'$
Uq)aYCbFQ(A`Z(a1Vp[[q,a q"!1U1B!!!!:
```

HTML-Mail

Mit dem Vormarsch des WWW wurden auch die E-Mails bunter. Statt Dateianhängen wurden vielfältige Formatierungsmöglichkeiten verlangt. Es lag nahe, HTML zur Kodierung von E-Mail zu verwenden. Mit MIME ist der Einbau sehr einfach möglich. Viele E-Mail-Clients erkennen E-Mail, die HTML enthält, und dekodieren diese problemlos. Wer eigene Programme schreibt, sollte entsprechende Nachrichten erkennen können. Im Usenet sind HTML-Mails verpönt und oft direkt verboten. Viele Newsreader können solche Nachrichten auch nicht darstellen.

Eine einfache HTML-Mail hat folgenden Aufbau:

```
MIME-Version: 1.0
Content-Type: multipart/alternative; boundary="----
= NextPart_000_0059_01BEA6E2.1A467F40"
This is a multi-part message in MIME format.

-----=_NextPart_000_0059_01BEA6E2.1A467F40
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding: 7bit

A simple HTML message.

-----=_NextPart_000_0059_01BEA6E2.1A467F40
Content-Type: text/html; charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
<HTML>
<HEAD>
<META content=3Dtext/html; charset=3Diso-8859-1 =
http-equiv=3DContent-Type>
<META content=3D'"MSHTML 4.72.3110.7"' name=3DGENERATOR>
</HEAD>
<BODY>
<DIV>A simple <STRONG>HTML</STRONG> message.</DIV>
<DIV> </DIV></BODY></HTML>
-----=_NextPart_000_0059_01BEA6E2.1A467F40-
```

Listing 2.4:
Eine kleine HTML-
E-Mail

Erkennbar sind drei wichtige Details:

- Das Multi-Part-Format ermöglicht die parallele Darstellung in mehreren Formaten, die Teile sind jeweils durch spezielle Zeichenfolgen (-----=_NextPart_ usw.) getrennt.

- ROT-13

Dieses Verfahren ist zwar bekannter, dient aber nicht der Kodierung, sondern der Verschlüsselung. Es ist extrem primitiv. Alle Buchstaben werden um 13 Positionen im Alphabet verschoben, aus »A« wird also »N« und aus »B« wird »O« usw. Sie erkennen ROT-13 an solchen Texten:

```
Guvf vf n fnzcyr bs n zrffntr rapbqrq hfvat EBG-13 rapbqvaf.
Orpnhfr bs gur fvzcyr angher bs gur rapelcgvba, vgf checbfr vf
abg frphevgl ohg gb cerirag nppvqragny ernqvaf.
```

Probleme mit E-Mail-Clients

Einige Probleme treten immer wieder auf. Sie sollten die Ursachen kennen, um bei eigenen Lösungen mit PHP nicht in dieselben Fallen zu treten. Das größte Problem sind Inkompatibilitäten zwischen Mail-Sendern und Mail-Empfängern. Die Ursache liegt oft im Zusammenreffen zweier Welten, Unix und Windows.

Probleme und deren Ursache

- Statt eines Anhangs ist nur kodierter Text zu sehen

Dies kann zwei Ursachen haben. Zum einen erkennt das Mailprogramm das Format nicht. Zum anderen würde es dies schon können, aber die Nachricht ist doppelt kodiert worden.

- Der Dateianhang ist unvollständig

Manche Nachrichten, die aus mehreren Teilen bestehen, werden stückchenweise übertragen und ein Teil der Übertragung ist noch nicht eingetroffen. Manchmal reicht auch der vorgesehene Platz in einer Mailbox nicht aus.

- Anstatt des Anhangs wird Text angezeigt, der offensichtlich aus dem Anhang stammt

Im Abschnitt zu MIME wurde bereits gesagt, dass das Sendeprogramm anhand der Daten entscheidet, ob die Kodierung in Base64 oder Quoted Printable erfolgt. Diese Entscheidung ist manchmal falsch. Vor allem Word-Dateien oder Excel-Tabellen müssen unbedingt Base64-kodiert sein, werden aber oft als Text erkannt.

- Der Text der Nachricht erscheint als Anhang, nicht als Text

Solche Nachrichten haben oft Anhänge der Art »ATT00001.TXT«. Die Ursachen sind vielfältig:

- Der Sender nutzt ein Textverarbeitungsprogramm wie MS Word. Auch wenn er den Text als ASCII speichert, wird er als Anhang versendet.
- Es wird eine Weiterleitungsfunktion (Forward) genutzt. Auch ohne Änderungen an der ursprünglichen Nachricht wird dar-

V
S
I
1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E

aus ein Anhang gebildet. Dieses Verhalten ist normal. Es kann aber unterdrückt werden, indem der Sender die MIME-Direktive Content-Disposition: inline anfügt. Leider ignorieren manche Mailprogramme dies.

- Manche Sender nutzen 8-Bit. Wenn ein Server auf dem Weg der E-Mail durch das Internet 7-Bit nutzt, wird er die Nachricht mit Quoted Printable wandeln. Er wird dies aber manchmal nicht für die ganze Nachricht tun, sondern nur für den Textblock, der wirklich 8-Bit-Zeichen enthält. So entstehen Anhänge. Diese Anhänge werden oft als »ATT00000.TXT« bezeichnet. Das Verhalten kann vermieden werden, indem der Sender einen üblichen Zeichensatz nutzt, beispielsweise den ISO-8859-1 (Western Alphabet ISO).

Header der E-Mail-Nachrichten

RFC 822

RFC 822 definiert die in E-Mail-Nachrichten zulässigen Header. Die folgende Aufstellung nennt alle dort definierten Header. Die Reihenfolge ist nur »empfohlen«. Halten Sie sich trotzdem daran, einige Mail-Clients könnten sonst Probleme machen.

Empfohlene Reihenfolge der Header

Richtig ist diese Reihenfolge: »Return-Path«, »Received«, »Date«, »From«, »Subject«, »Sender«, »To«, »cc« usw.

Die folgende Aufstellung enthält alle Header, der Punkt vor der Beschreibung gehört nicht zur Syntax. Angaben in eckigen Klammern [] sind optional:

- Return-path : *route-addr*

Pfad zurück zum Absender.

- Received : *further-parameters*

Für *further-parameters* setzen Sie einen oder mehrere der folgenden Parameter, jeweils zeilenweise getrennt, ein:

- [from *domain*]

Name des sendenden Computers

- [by *domain*]

Name des empfangenen Computers

- [via *atom*]

Physischer Pfad zu E-Mail.

- *(with *atom*)

Protokoll für die Übertragung.

- [id *msg-id*]
Nachrichten-ID des Empfängers.
- [for *addr-spec*]
Formular.
- ; date-time
date-time ist die Empfangszeit.
- [Reply-To : *address*]
address muss mindestens ein Mal angegeben werden und kann aus folgenden Elementen bestehen:
 - From : *mailbox*.
Absenderadresse (Autor).
 - Sender : *mailbox*
Sendeeinrichtung, kann mit dem Autor identisch sein, kann aber auch seinen Senderserver bezeichnen und wird dann mit from kombiniert.
 - [Resent-Reply-To : *address*]
- Resent-From : *mailbox*
Weiterleitungsinformation. Eine alternative Angabe dazu kann aus zwei Elementen bestehen:
 - Resent-Sender : *mailbox*
Resent-From : *mailbox*
Weiterleitungsinformation mit Senderkennung und Absenderdaten.
- Date : *date-time*
Sendedatum der Nachricht.
- Resent-Date : *date-time*
Zeitpunkt einer Weiterleitung.
- To : *address*
Primäre Empfängeradresse. Mindestens eine Angabe muss hier stehen. Ergänzend können weitere Header folgen:
 - Resent-To : *address*
Empfänger einer Weiterleitung.
 - cc : *address*
Kopie an weiteren Empfänger (carbon copy)

V
S
I
1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E

- Resent-cc : *address*
Kopie einer Weiterleitung.
 - bcc : *address*
Verdeckte Kopie (blind carbon copy)
 - Resent-bcc : *address*
Verdeckte Kopie einer Weiterleitung.
 - Message-ID : *msg-id*
Eindeutige ID des Empfängers.
 - Resent-Message-ID : *msg-id*
Eindeutige ID einer Weiterleitung.
 - In-Reply-To : **(phrase | msg-id)*
Bezugnahme einer Antwort auf eine andere E-Mail, die Zuordnung erfolgt durch eine Phrase (beispielsweise den Betreff) oder die Message-ID.
 - References : **(phrase | msg-id)*
Referenz auf eine andere E-Mail, findet vor allem bei Mailinglisten Anwendung. Die Zuordnung erfolgt durch eine Phrase (beispielsweise den Betreff) oder die Message-ID.
 - Keywords : *phrase*
Schlüsselworte; wird nur selten verwendet.
 - Subject : *text*
Betreff der Nachricht, sollte unbedingt verwendet werden.
 - Comments : **text*
Kommentare.
 - Encrypted : *2word*
Hinweis auf eine bestehende Verschlüsselung des Nachrichten-Körpers.
 - extension-field
Erweiterung; nicht offiziell definiert.
 - user-defined-field
Nutzerspezifische Erweiterungen, beispielsweise Hinweise auf Priorität.
- < *addr-spec* >
Eindeutige ID der Nachricht.

Das folgende Beispiel zeigt, wie die Header einer E-Mail aussehen können. Die Mail entstammt übrigens der deutschen PHP-Mailingliste. Die Header sind fett hervorgehoben, die optionalen Attribute kursiv.

Wegen des begrenzten Platzes pro Zeile wurden einige Umbrüche eingefügt, diese sind mit _ gekennzeichnet. Generell beginnt auf jeder Zeile ein neuer Header, wenn der Text in der ersten Spalte startet.



Gut zu erkennen sind die vielen zusätzlichen, nicht RFC 822-konformen Header und der komplexe Pfad, den diese E-Mail genommen hat (sicher auch provoziert durch eine Weiterleitung):

```
Return-path: <php-admin@solix.wiso.Uni-Koeln.DE>
Envelope-to: joerg.krause.net@comzept.de
Delivery-date: Sat, 22 Apr 2000 00:31:15 +0200
Received: from [208.49.167.126] (helo=mb3.mailbank.com)
    by mx01.kundenserver.de with esmtp (Exim 2.12 #2)
    id 12ilw7-00007M-00
    for joerg@comzept.de; Sat, 22 Apr 2000 00:30:47 +0200
Received: from infosoc.uni-koeln.de (solix.wiso.Uni-Koeln.DE)
    by mb3.mailbank.com (8.9.1/8.9.1) with SMTP id PAA29717
    for <joerg@krause.net>; Fri, 21 Apr 2000 15:30:04 -0700
Received: (qmail 14450 invoked from network); 21 Apr 2000
    22:26:17 -0000
Received: from solix.wiso.uni-koeln.de (daemon@134.95.183.82)
    by solix.wiso.uni-koeln.de with SMTP; 21 Apr 2000
    22:26:17 -0000
Received: from twdc.de (qmailr@[212.172.92.2])
    by infosoc.uni-koeln.de (8.8.8/8.8.8) with SMTP id _
    AAA14428
    for <php@solix.wiso.Uni-Koeln.DE>; Sat, 22 Apr 2000
    00:26:10 +0200
Received: (qmail 20861 invoked from network); 21 Apr 2000
    19:39:42 -0000
Received: from dialin126-nt.pg7.frankfurt.nikoma.de _
    (HELO claudia) (213.54.38.126)
    by 212.172.92.3 with SMTP; 21 Apr 2000 19:39:42 -0000
Message-ID: <001501bfabd2$188c6840$7e2636d5@claudia>
From: "Thorsten Habich" <Antares@www-surfen.de>
To: <php@solix.wiso.Uni-Koeln.DE>
References: <004501bfabac$e9c52800$d47c353e@n4n8u9> _
    <39009410.44C4B172@kawo2.rwth-aachen.de>
Subject: Re: [php] Live-Counter
Date: Fri, 21 Apr 2000 20:50:37 +0200
Organization: X-press
MIME-Version: 1.0
Content-Type: text/plain; charset="iso-8859-1"
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 5.00.2014.211
```

Listing 2.5:
Header einer
»gewöhnlichen« E-
Mail (aus einer
Mailingliste)

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E


```

X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2014.211
Reply-To: php@solix.wiso.Uni-Koeln.DE
Sender: php-admin@solix.wiso.Uni-Koeln.DE
Errors-To: php-admin@solix.wiso.Uni-Koeln.DE
X-Mailman-Version: 1.0b8
Precedence: bulk
List-Id: deutschsprachige PHP-Mailingliste _
        <php.infosoc.uni-koeln.de>
X-BeenThere: php@infosoc.uni-koeln.de
Content-Transfer-Encoding: quoted-printable
X-MIME-Autoconverted: from 8bit to quoted-printable by _
        mb3.mailbank.com id PAA29717

```

RFC 977 RFC 2980

2.2.5 Network News Transfer Protocol (NNTP)

NNTP ist ein sehr altes Protokoll – Nachrichtengruppen gab es lange vor der Eroberung des Internets durch das WWW. Es wurde in RFC 977 vom Februar 1986 definiert. Seit August 1996 gibt es einen Draft mit einer Reihe von Erweiterungen, die zwischenzeitlich ohne Standard eingeführt wurden. Dieser Draft mündete im Dezember 2000 in die RFC 2980, die die Erweiterungen von NNTP unter expliziter Bezugnahme auf RFC 977 beschreibt.

Arbeitsweise

Das Protokoll wird durch den Austausch von Texten zwischen Server und Client bestimmt. Die rein ASCII-basierte Übermittlung erspart die typischen Netzprobleme mit Binärdaten. Der Server arbeitet nicht statuslos, sondern führt einen so genannten Message-Pointer. Aus diesem Grunde ist eine Anmeldung erforderlich und es sollte auch eine Abmeldung erfolgen.

Der Client muss die Kommunikation aufnehmen und sollte in der Lage sein, alle denkbaren Antworten des Servers zumindest soweit zu bearbeiten, dass kein Blockieren im Protokoll entsteht.

Serverantworten

Auf jede Anfrage wird der Server zunächst mit einem dreistelligen Zahlencode reagieren. Die folgende Tabelle zeigt die möglichen Antwortklassen:

Tabelle 2.21:
Antwortklassen in
NNTP

Code	Beschreibung
1xx	Informationen
2xx	Kommando korrekt
3xx	Kommando soweit korrekt, erwarte Daten

Code	Beschreibung
4xx	Kommando war korrekt, konnte aber nicht ausgeführt werden
5xx	Kommando unbekannt oder Fehler

Die nächste Stelle sagt etwas über die Kategorie:

Code	Beschreibung
x0x	Verbindung, Setup und sonstige Nachrichten
x1x	Newsgroupauswahl
x2x	Artikelauswahl
x3x	Distributionsfunktionen
x4x	Senden von Artikeln
x8x	Erweiterungen, die nicht standardisiert sind.
x9x	Debug-Ausgaben

Tabelle 2.22:
Kategorie innerhalb
des Antwortcodes

Beispielsitzung

Der Client fragt nach den Gruppen auf dem Server:

```
LIST
```

Der Server antwortet zunächst mit dem Code 215, dass die Liste mit den Gruppen folgt und sendet dann Zeile für Zeile diese Gruppen:

```
groupname last first p
```

Dabei ist groupname der Gruppenname, beispielsweise microsoft.iis.de ist die letzte, also neueste Nachricht in der Gruppe, analog ist first die erste verfügbare Nachrichtennummer. p kann »y« oder »n« sein und gibt an, ob Benutzer in der Gruppe Nachrichten absetzen dürfen oder nicht. Die Sendung wird durch das Senden einer Zeile, die nur einen Punkt enthält, beendet.

Der Client wechselt in eine der Gruppen durch Senden des GROUP-Kommandos. Als Parameter gibt er den Namen der Gruppe an. Der Server bestätigt mit folgender Zeile:

```
211 article_count first last groupname
```

Mit dem folgenden Befehl wird die Nummer der Nachricht aufgerufen:

```
STAT MsgNr
```

Mit dem Kommando HEAD wird der Kopf der Nachricht und mit BODY der Inhalt der Nachricht bezeichnet. Durch das Kommando NEXT kann die nächste Nachricht gelesen werden.

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Das Senden von Artikeln

Nach dem Senden von POST wird der Server zunächst mitteilen, ob er Senden akzeptieren kann. Der Antwort-Code kann einer der folgenden sein:

*Tabelle 2.23:
Mögliche Antwort-
codes nach POST*

Code	Beschreibung
240	Artikel wurde bereits gesendet.
340	Aufforderung zum Senden. Der Artikel soll mit der Folge <CR-LF>.<CR-LF> beendet werden.
440	Posten von Artikeln ist nicht erlaubt.
441	Das Posten ist misslungen.

RFC 850
RFC 1036

RFC 850 gilt für das Format einer Nachricht. Speziell für die Usenet News Artikel gilt die RFC 1036.

Beenden der Verbindung

Auf das Kommando QUIT sendet der Server die Antwort 205. Der Vorgang ist damit abgeschlossen und bei späteren Zugriffen ist eine erneute Anmeldung notwendig.

Threadverfolgung

Threads sind Diskussionsbäume, die durch Erwiderung auf Artikel entstehen. Im Artikel wird durch Belegen des Feldes References: die Message-ID abgelegt, auf die der Artikel sich bezieht. Es gibt also Rückwärts-, aber keine Vorwärtsbezüge.

2.3 Vorbereitung der Installation

Mit den theoretischen Grundlagen gerüstet, können Sie nun die Installation beginnen. Wenn Sie noch keine Technik vorbereitet haben, hilft Ihnen dieser Abschnitt, die richtigen Komponenten zusammen zustellen.

2.3.1 Beschaffung der Komponenten

**Was wird
gebraucht?**

Die Vorbereitung zur Installation besteht im Wesentlichen aus zwei Teilen:

- Beschaffung der Hardware
- Beschaffung der Software

Die Beschaffung der Hardware dürfte in den meisten Fällen kein Problem bereiten, da sich PHP auf fast jedem Computer entwickeln lässt.

Planen Sie eine Neuanschaffung, sollte ein Entwicklungssystem folgende Mindestanforderungen erfüllen:

- Intel- oder AMD-Prozessor mit 400 MHz oder mehr
- 128 MB Hauptspeicher
- 10 GB Festplatte (wünschenswert sind UW-SCSI-Systeme)

Die Software hängt vom gewählten Betriebssystem ab. Die folgenden Komponenten sind also entsprechend aufeinander abzustimmen:

- Betriebssystem (Windows, Windows NT/2000, Linux, BSD-Unix, Solaris oder ein anderes kommerzielles Unix).
- Webserver (Internet Information Server, Apache, Xitami).
- PHP in einer zum Betriebssystem passenden Binärdistribution (andernfalls müssten Sie den Quellcode zuvor mit einem C-Compiler übersetzen).
- MySQL oder eine andere SQL-Datenbank, die von PHP unterstützt wird oder über ODBC-Treiber verfügt.

Die folgenden Bausteine finden Sie auf der CD:

- PHP 4 in der Binärdistribution (Version 4.0.6) für Windows
- MySQL 3.32 in der Binärdistribution für Windows
- PHP 4 und MySQL 3.23 als Quellcode für Unix und Windows
- Apache Webserver in der Version 1.3.20 als Binärdistribution
- Apache Webserver für Linux als Quellcode

2.4 Installation unter Windows NT/2000

Wenn PHP auf dem Server des Providers läuft, kommt meist Linux zum Einsatz. Für den schnellen Einstieg in die Programmierung zu Hause eignet sich Windows ebenso.

2.4.1 WAMP

Die Installation unter Windows ist relativ einfach und unproblematisch. Der Einsatz des Apache-Webservers ist eher eine Glaubensfrage. Praktisch steht er ebenso kostenlos zur Verfügung wie der IIS oder PWS. Sollte sich Microsoft die Verteilungspolitik anders überlegen, wäre der Apache allerdings ein ebenbürtiger und ausgereifter Ersatz für den IIS.

WAMP →
Windows
Apache
MySQL
PHP

Windows NT 4
Win 95
Win 98
Windows 2000
Windows XP

Windows vorbereiten

Grundsätzliche Gründe, die vorgeschlagene Kombination aus Apache, PHP und MySQL nicht zu installieren, gibt es nicht. Wenn ein Produktionsserver⁵ aufgebaut wird, ist ein nacktes Windows zu empfehlen. Ältere Versionen der zu installierenden Software sollten auf jeden Fall sauber deinstalliert werden. Service Packs unter NT sind nicht relevant, im Hinblick auf die Systemsicherheit sollten aber die neuesten Service Packs oder Patches installiert sein.

Apache konfigurieren

Die Installation des Apache ist einfach und ohne weiteres auszuführen. Laden Sie sich die Win32-Binaries von <http://www.apache.org>. Um mit anderen Informationsquellen arbeiten zu können, ist eine Installation im Standardverzeichnis C:\APACHE anzuraten.

http.conf

Öffnen Sie nun die Datei HTTP.CONF im Verzeichnis \APACHE\CONF. Suchen Sie die folgende Zeile:

```
DirectoryIndex
```

Ergänzen Sie nun diese Zeile um die folgenden Einträge:

```
index.php3  
index.php4  
index.php
```



Hinweis

Die Dateierweiterung PHP3 wird üblicherweise für PHP 3 verwendet, während PHP 4 ohne Ziffer in der Endung arbeitet. Prinzipiell sind Sie hier in der Wahl der Endung frei. Die Verknüpfung beider Endungen mit PHP 4 ist sinnvoll, wenn Sie viele Skripte aus dem Web beschaffen und nicht alles umbenennen möchten. Wenn Sie PHP 3 und PHP 4 parallel verwenden, sollten Sie als Endung für PHP 4 PHP4 nutzen.

Damit akzeptiert Apache nun auch PHP-Dokumente als Standarddokument. Die Zuordnung der Dateierweiterung zum PHP-Parser ist bereits vorbereitet. Sie müssen lediglich den Kommentar vor der folgenden Zeile entfernen:

```
AddType application/x-httpd-php3 .php3
```

Fügen Sie außerdem auch die folgende Zeile hinzu, damit Apache auch Dateien mit der verkürzten Endung .PHP annimmt:

```
AddType application/x-httpd-php .php
```

Der letzte Schritt verknüpft nun die akzeptierten Dateierweiterungen mit dem PHP-Parser selbst. Fügen Sie die folgende Zeile hinzu:

⁵ Das sind die Server, die später »ans Netz gehen«.

```
Action application/x-httpd-php /cgi-bin/php.exe
```

Voraussetzung ist natürlich, dass PHP auch im virtuellen Verzeichnis /CGI-BIN installiert wurde. Die Darstellung geht auch davon aus, dass es sich um die CGI-Version handelt.

Installieren Sie Apache unter Windows NT 4/2000 oder XP als Dienst. Dies ist bei einem Produktionsserver unbedingt notwendig, damit bereits vor dem Einloggen der Webserver startet. Auf einem Entwicklungssystem kann auch gut Windows 9x oder Windows Me zum Einsatz kommen. Hier gibt es keine Dienste, installieren Sie den Aufruf des Apache deshalb im Autostart-Ordner.

Zum Test geben Sie `http://localhost` in Ihrem Browser ein – der Apache sollte sich melden und Sie können mit der Installation fortfahren.

PHP 3

Die aktuellste Version der Win32-Binaries bekommen Sie unter `http://www.php.net`. Standardmäßig wird in `C:\PHP3` installiert. Kopieren Sie nun die `PHP3.EXE` und die `.DLL`-Dateien nach `\CGI-BIN`. Für die Installation von MySQL benötigen Sie:

- `LIBMYSQL.DLL`
- `PHP3_MYSQL.DLL`

Dann bearbeiten Sie die Datei `PHP3.INI`. Wenn sie noch nicht vorhanden ist, kopieren Sie die Quelle `PHP3.INI-DIST` ins Windows-Verzeichnis und benennen sie in `PHP3.INI` um. Eine Anleitung zur Konfiguration von PHP mit Hilfe dieser Datei finden Sie in ➡ Abschnitt 2.6 *PHP konfigurieren* ab Seite 119.

Der Einsatz von PHP 3 ist heute nicht mehr zu empfehlen. Die Beschäftigung damit ist nur sinnvoll, wenn Sie gelegentlich Skripte für Kunden entwickeln, deren Server – aus welchen Gründen auch immer – nicht mit PHP 4 arbeiten. Haben Sie solche Bedingungen nicht, vergessen Sie PHP 3.

PHP 4

Die Binärdateien für PHP 4 enthalten zwei Versionen der Software: eine für die CGI-Version, die andere für ISAPI. Auf ISAPI geht der folgende Abschnitt ein. Die CGI-Version kann sowohl für den Apache (wie oben beschrieben) als auch für den IIS (siehe auch nächster Abschnitt) verwendet werden.

Der wesentliche Unterschied zu PHP 3 ist die Tatsache, dass das MySQL-Modul nun bereits hineinkompiliert wurde. Das ist wichtig zu wissen, denn die entsprechende DLL wird nicht mehr benötigt. In der `PHP.INI` von PHP 4 gibt es dennoch eine DLL-Extension dafür – deren



Aktivierung führt aber zum Absturz. Mehr zu php.ini und den Besonderheiten der Einrichtung finden Sie in ➡Abschnitt 2.6 *PHP konfigurieren* ab Seite 119.

2.4.2 WIMP

**WIMP →
Windows
IIS
MySQL
PHP**

Eine besonders einfache Kombination stellt die Installation von PHP auf einem Windows mit dem mitgelieferten IIS (oder PWS, Personal Web Server) dar. Als Produktionsserver dürfte die Kombination nicht zu empfehlen sein (außer für extrem kleine Sites), als Entwicklungssystem hat die Variante einige Vorteile. Außer der Investition in das Betriebssystem sind alle Komponenten frei. Der IIS ist im Gegensatz zum Apache mit einer grafischen Bedienoberfläche ausgestattet, die es auch Anfängern erlaubt, innerhalb kürzester Zeit ein System lauffähig zu bekommen. Auch ohne Systemkenntnisse ist der IIS sehr schnell betriebsbereit. PHP 3 ist zwar nur als CGI-Programm verfügbar, dies erleichtert den Umgang jedoch gerade in der Anfangszeit erheblich. Die mangelnde Performance ist auf einem Einzelplatzsystem völlig unerheblich. PHP 4 steht auch als ISAPI-Erweiterung zur Verfügung, hier gibt es nicht einmal diesen Nachteil. Wenn Sie es gewohnt sind, mit Windows zu arbeiten, sollten Sie diese Kombination ernsthaft in Erwägung ziehen. Der nächste Abschnitt beschreibt die Installation der ISAPI-Module unter Windows 2000.

PHP als ISAPI-Modul im IIS 5

**PHP 4 und
Windows 2000**

Wenn Sie mit dem IIS arbeiten, können Sie PHP als ISAPI-Modul betreiben. Dies ist ähnlich leistungsfähig wie das PHP-Modul für den Apache unter Linux. Leider war auch die letzte vor Drucklegung getestete PHP 4-Version, PHP 4.0.6 unter ISAPI nicht stabil. Versuchen Sie, über <http://www.php.net> eine neuere Version zu erhalten.

Zur Installation gehen Sie folgendermaßen vor:

- Entpacken Sie die ZIP-Datei
- Kopieren Sie die *.DLL und *.EXE-Dateien nach
`%windir%\system32\inetsrv\`
- Öffnen Sie die Verwaltung des IIS in der Management-Konsole
- Wählen Sie das Standardverzeichnis und öffnen Sie den Dialog EIGENSCHAFTEN.
- Wechseln Sie zur Registerkarte BASISVERZEICHNIS und klicken dort auf KONFIGURATION.
- Im folgenden Dialog wählen Sie ANWENDUNGSKONFIGURATION und fügen dort zwei Einträge der Liste hinzu, einen für die En-

nung PHP und einen für PHP3. Dabei geben Sie den Pfad zu der Datei PHP.EXE ein, wenn Sie mit der CGI-Version arbeiten möchten. Wenn Sie ISAPI bevorzugen, wird der Pfad zu PHP4ISAPI.DLL angegeben.

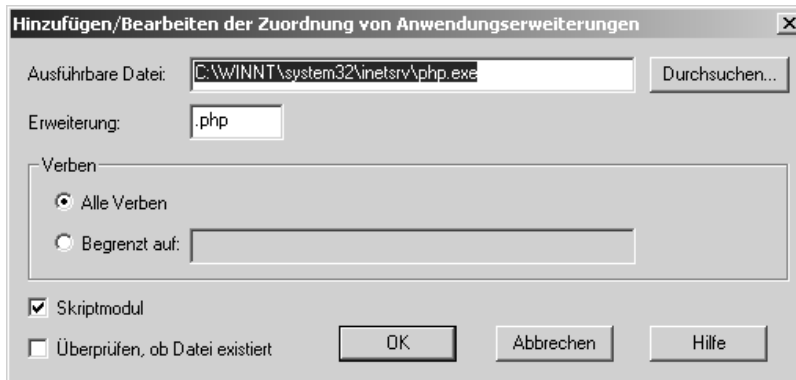


Abbildung 2.1:
Verknüpfung der
CGI-Anwendung

Für die ISAPI-Version können Sie außerdem noch ein Filter installieren. Filter werden vor dem Parser ausgeführt. Damit kann auch die integrierte Authentifizierung des Webservers verwendet werden. CGI kann dies nicht. Die Verwendung ist auch bei ISAPI optional – selbst wenn die mitgelieferte Kurzanleitung etwas anderes suggeriert.

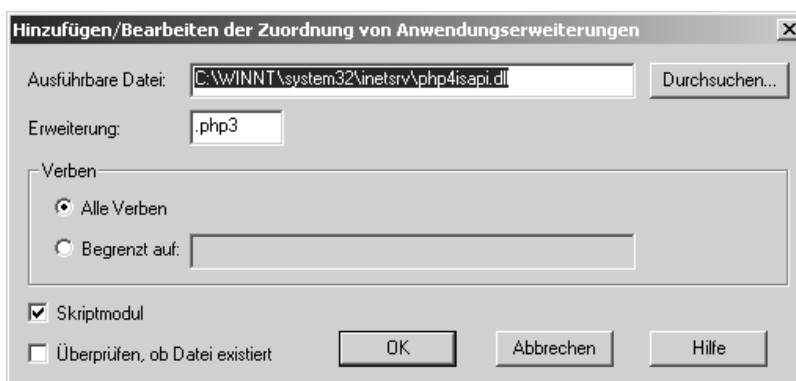


Abbildung 2.2:
Verknüpfung mit der
ISAPI-Version

Die Einrichtung des Filter erfolgt über die Registerkarte ISAPI-FILTER des Dialogs EIGENSCHAFTEN. Fügen Sie dort ein Filter mit dem Namen »PHP« ein. Unter PFAD wird auf die Datei PHP4ISAPI.DLL verwiesen.

Jetzt muss der Webserver neu gestartet werden, entweder über DIENSTE in der Systemsteuerung oder das entsprechende Konsolenkommando:

```
C:>net stop w3svc  
C:>net start w3svc
```


Falls das ISAPI-Modul abstürzt, genügt es ebenfalls, den Webserver neu zu starten. Solange der Zustand nicht stabil ist, sollten Sie ohnehin auf den Einsatz Webserver, der im Internet frei verfügbar ist, verzichten.

PHP und der PWS

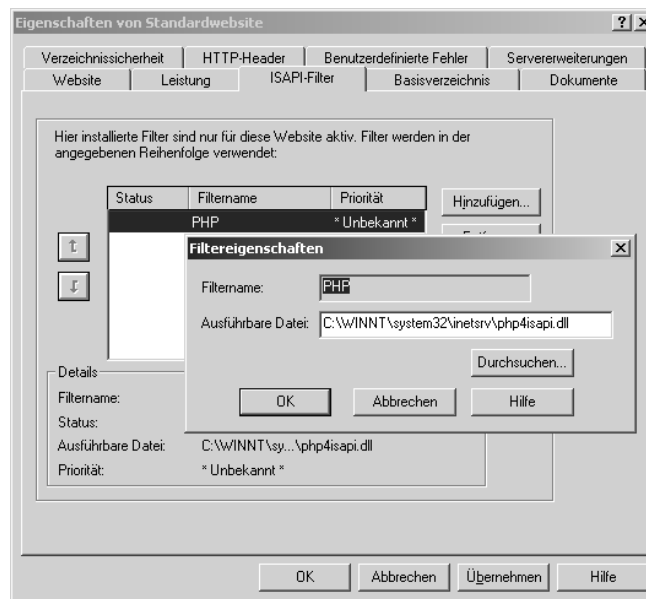
Der PWS (*Personal Web Server*) ist eine primitive Version des IIS und wird mit Windows 9x geliefert. Die Einrichtung müssen Sie hier in der Registrierung vornehmen. Gehen Sie folgendermaßen vor:

1. Installieren Sie die Dateien, wie oben bereits beschrieben.
2. Bearbeiten Sie die Datei PWS-PHP4.REG so, dass der Inhalt Ihren Systembedingungen entspricht:

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\
w3svc\parameters\Script Map]
".php"="C:\\Program Files\\PHP\\php4isapi.dll"
```

3. Im PWS-Manager aktivieren Sie für das Verzeichnis, in dem PHP-Skripte verarbeitet werden soll, die Ausführungsberechtigung. Dazu wird das Kontrollkästchen AUSFÜHREN aktiviert.

Abbildung 2.3:
Einrichten eines
ISAPI-Filters



2.4.3 WXMP

Xitami ist als Webserver unter Windows recht beliebt. Er ist wesentlich problemloser zu installieren und einzurichten als der Apache Webserver und weniger komplex als der IIS. Windows 9x-Nutzer, die ohnehin keinen IIS haben, sollten Xitami als Webserver ernsthaft in Erwägung ziehen. Xitami erscheint in einer Windows-konformen Oberfläche.

WXMP →
Windows
Xitami
MySQL
PHP

Die für die Installation benötigten Dateien finden Sie unter folgenden Adressen:

- Xitami

<http://www.imatix.com>

- MySQL

<http://www.mysql.com>

- PHP

<http://www.php.net>

Die folgenden Programme sollten in das Root-Verzeichnis installiert werden, also nach »C:\«. Das ist kein Zwang, spart aber vor allem Anfängern manchen Ärger. Windows 9x/ME/XP/NT oder 2000 sollten vorhanden sein, Voraussetzung für Win95-User ist die Winsock 2. Zu finden sind aktuelle Versionen für Windows 95 unter der folgenden Adresse:

<http://www.microsoft.com/windows95/downloads/>

Windows NT braucht den Service Pack 2 (oder neuer). Gelegentlich kommt es zu Problemen mit ODBC-Treibern. Installieren Sie unbedingt die neuesten Versionen, die verfügbar sind.

Xitami wird mit dem mitgelieferten Setup-Programm SETUP.EXE installiert. Damit PHP-Skripte erkannt und ausgeführt werden, muss die Dateierweiterung PHP, PHP3 oder PHP4 (wenn gewünscht) mit der entsprechenden PHP.EXE verknüpft werden. Dazu wird in der Datei »xitami.cfg«, die im Verzeichnis C:\XITAMI\ (Sektion [Filter]) zu finden ist, mindestens die folgende Zeile eingefügt:

Xitami
konfigurieren

```
php = php.exe
```

Es ist am einfachsten, wenn Sie alle Dateien der Binärdistribution von PHP in das Xitami-Stammverzeichnis kopieren. Andernfalls geben Sie in der »xitami.cfg« den kompletten Pfad zu den Dateien mit an und legen einen Suchpfad auf das PHP-Verzeichnis.

Jetzt wird die Datei PHP.INI angepasst. Ausführliche Informationen dazu finden Sie im ➡ Abschnitt 2.6.1 *Die Datei php.ini* ab Seite 119.

php.ini anpassen

Zusätzlich tragen Sie die Erweiterungen ein, wenn diese gewünscht sind:

```
extension_dir = ./c:\xitami
extension = php_gd.dll
```

Bearbeiten Sie nun die Datei PHP_IIS_REG.INF. Dort muss der korrekte Pfad zu PHP.EXE an allen Stellen stehen, wo Pfadangaben auftauchen. Installieren Sie nun die INF-Datei. Dazu klicken Sie mit der rechten Maustaste darauf und starten die Installation mit INSTALLIEREN.

Wenn Sie nun eine Testdatei nach CGI-BIN kopieren, können Sie Ihren Xitami mit PHP ausprobieren:

```
http://localhost/cgi-bin/test.php
```

MySQL wird wie üblich installiert. Mehr Informationen zu MySQL sind in ➔ Abschnitt 6.1 *Installation von MySQL* ab Seite 457 zu finden.



Um MySQL unter PHP 3 nutzen zu können, muss die entsprechende DLL aktiviert werden. Entfernen Sie das Kommentarzeichen vor der Zeile:

```
;Windows Extensions
extension=php3_mysql.dll
```



PHP 4 wird bereits mit einkompilierter MySQL-Unterstützung geliefert. Hier sind die entsprechenden Angaben nicht notwendig.

2.5 Installation unter Linux

LAMP

Linux
Apache
MySQL
PHP

Linux ist sicher die typischste Plattform für PHP. Die Installation setzt in der Regel eine Übersetzung des Quellcodes voraus. Damit erwirbt man auch die Freiheit, bestimmte Module zu nutzen, die standardmäßig nicht enthalten sind. Die folgende Beschreibung zeigt die Installation unter Linux und das Zusammenspiel mit dem Apache-Webserver.

2.5.1 Vorhandenes System nutzen

Der Wunsch nach einer leistungsfähigen Skriptsprache kommt oft erst nach der erfolgreichen Installation eines Webserver. Wenn Sie bereits Erfahrung im Umgang mit Linux haben und der Apache-Webserver stabil läuft, nimmt die Installation von PHP nur noch wenig Zeit in Anspruch.

Prinzipiell kann PHP als Skriptmodul (CGI) oder als Modul im Apache betrieben werden. Die Integration in den Apache bietet einen deutlichen Performancevorteil. Darüber hinaus sind einige servernahe Funktionen nutzbar. Nachteilig ist die Abhängigkeit so entstandener

Skripte (wenn Sie diese Funktionen auch nutzen) von der Konfiguration. Die Portierbarkeit nach Windows oder auf andere Unixe mit anderen Webservern ist eingeschränkt.

Vorbemerkung

Die folgende Anleitung geht davon aus, dass Sie selbst alle Komponenten kompilieren. Auf spezielle Module und Erweiterung wird nicht eingegangen, weil diese jedes für sich bestimmte Probleme verursachen können. Sie sollten erst ein »nacktes« System lauffähig bekommen und dann mit zusätzlichen Bibliotheken experimentieren. Voraussetzung ist außerdem, das Linux selbst einwandfrei läuft und Sie halbwegs damit umgehen können.

Wenn schon Übersetzungsversuche gestartet wurden, räumen Sie alle Verzeichnisse folgendermaßen auf:

- Zuerst sind die Dateien CONFIG.STATUS und CONFIG.CACHE zu löschen.
- Dann ist folgender Aufruf auszuführen:

```
$ make clean
$ ./configure
```

Linux vorbereiten

Das gesamte Paket besteht aus dem Apache Webserver, PHP und MySQL. Diese drei Programme sollten als Tarball (*.TAR.GZ) erst einmal beschafft und abgelegt werden, beispielsweise unter diesem Pfad:

```
/usr/local/src/lampe
```

Wenn ein Linux neu installiert wird, kommt die Frage nach verschiedenen Paketen. Folgende werden gebraucht: *flex*, *bison*, *xdev* und *libpng*.

Es gibt noch ein paar Zusatzmodule, die man nicht unbedingt haben muss, wenn der Linux-Server als Entwicklungssystem installiert wird. Dazu gehört WebDAV (Autorenverwaltung) und SSL (Verschlüsselung). Darauf wird hier nicht eingegangen. Wichtiger ist dagegen die GD-Bibliothek, die Bilder dynamisch erzeugt. Diese wird auch im Buch verwendet und wird hier mit integriert. Auch diese liegt als Tarball vor und wird im Quellverzeichnis /LAMPE abgelegt. Aktuell ist GD-1.8.4.TAR.GZ. Benötigt wird auch die Komprimierungsbibliothek Zlib 1.1.3 (ZLIB.TAR.GZ) sowie IJG JPEG Software Version 6b (JPEGSRC.V6B.TAR.GZ) und Freetype Version 1.3.1 (FREETYPE-1.3.1.TAR.GZ).

Erweiterungen

Installationsschritte

Die Reihenfolge ist wichtig, wenn der Prozess abbricht und Sie erneut starten, räumen Sie vorher alles auf und beginnen dann wieder von vorn – auf keinen Fall mittendrin. Außerdem sollten Sie daran denken, sich als `ROOT` anzumelden, damit es keine Probleme mit unzureichenden Rechten gibt.

Zunächst ist die gesamte Software zu entpacken:

Entpacken

```
$ cd /usr/local/src/lampe
$ tar -xzf tarballs/apache_1.3.20.tar.gz
$ tar -xzf tarballs/freetype-1.3.1.tar.gz
$ tar -xzf tarballs/gd-1.8.4.tar.gz
$ tar -xzf tarballs/mysql-3.23.38.tar.gz
$ tar -xzf tarballs/php-4.0.6.tar.gz
$ tar -xzf tarballs/zlib.tar.gz
$ tar -xzf tarballs/jpegsrvc.v6b.tar.gz
```

Nun wird als erstes MySQL installiert. Das Datenverzeichnis ist getrennt vom Programmverzeichnis. Am Ende wird der Server gestartet. Wenn Sie einen öffentlichen Server haben, sollte das Konto `root` des MySQL-Servers (das hat nichts dem Root des Linux zu tun) mit einem Kennwort versehen werden. Auf einem privaten Testrechner ist es eher lästig.

MySQL installieren

```
$ cd /usr/local/src/lamp/mysql-3.23.38/
$ ./configure --prefix=/usr/local/mysql/3.23.38
  --localstatedir=/var/mysql/data
$ make
$ make install
$ ln -s /usr/local/mysql/3.23.38 /usr/local/mysql/current
$ mkdir /var/mysql
$ mkdir /var/mysql/data
$ scripts/mysql_install_db
$ /usr/local/mysql/current/bin/safe_mysqld &
```

Das Root-Kennwort wird folgendermaßen gesetzt:

```
$ /usr/local/mysql/current/bin/mysqladmin -u root password 'xxx'
```

Wenn Sie möchten, kann die Startup-Datei noch an die entsprechende Stelle kopiert werden, damit beim nächsten Hochfahren der Serverstart automatisch erfolgt. Bei SUSE sieht das folgendermaßen aus:

```
$ cp support-files/mysql.server /etc/rc.d/init.d
$ chmod 744 /etc/rc.d/mysql.server
$ cd /etc/rc.d/rc2.d
$ ln -s ../init.d/mysql.server S20mysql.server
$ ln -s ../init.d/mysql.server K20mysql.server
```

Apache

Zunächst wird der Apache kompiliert. Es ist empfehlenswert, eine DSO-Version zu erstellen, mit der Zusatzmodule wie PHP nicht fest

einkompiliert, sondern vom Server dynamisch zugeladen werden. Das hat in der Anwendung den Effekt, dass man beim Update von PHP nur noch PHP neu übersetzen muss, und nicht mehr den ganzen Apache. Die Übersetzung des Webservers ist sehr einfach, die Installation schließt sich daran an:

```
$ cd /usr/local/src/lamp/apache_1.3.20
$ make
$ make install
$ ln -s /usr/local/apache/1.3.20 /usr/local/apache/current
```

Zunächst müssen die Bibliotheken vorbereitet werden, damit die Integration in PHP später funktioniert. Wenn Sie hier Probleme haben, lassen Sie Bibliotheken weg und übersetzen PHP alleine. Funktioniert dies, versuchen Sie es wieder mit den Bibliotheken.

**Zusatzfunktionen
für PHP 4**

Die Installation erfolgt unter /usr/local. Freetype wird bei der Installation der GD-Bibliothek gebraucht. Geben Sie nun folgendes ein:

**Bibliothek:
Freetype**

```
$ cd /usr/local/src/lamp/freetype-1.3.1
$ make CFG="--prefix=/usr/local"
$ ./configure --prefix=/usr/local
$ make
$ make install
```

Die Bibliothek benötigt Fonts, die mit dem X-Paket *xdevel* installiert werden. Falls Fehler auftreten, konsultieren Sie YAST, um den Status der Pakete abzufragen.

Auch die *zlib* wird unter /usr/local installiert. Dann geben Sie folgendes ein:

Bibliothek: Zlib

```
$ cd /usr/local/src/lamp/zlib-1.1.3
$ ./configure
$ make
$ make install
```

Die *libjpeg* wird auch unter /usr/local installiert. Dazu gehen Sie folgendermaßen vor:

**Bibliothek: IJG
JPEG software**

```
$ cd /usr/local/src/lamp/jpeg-6b
$ ./configure \ --enable-shared \ --enable-static \
               --prefix=/usr/local
$ make
$ make test
$ make install
```

Die GD-Bibliothek ist etwas aufwändiger. Hier müssen in der Make-Datei die Zeilen CFLAGS und LIBS angepasst werden. Dort sollten Sie die vorgegebenen Zeilen kommentieren und die Alternativen auskommentieren. Bei INCLUDEDIRS und LIBDIRS ist jeweils ein Zusatz in der folgenden Form anzugeben:

GD

```
-I/usr/local/include/freetype2
-L/usr/local/lib
```

Außerdem kontrollieren Sie folgende Zeilen:

```
INSTALL_LIB=/usr/local/lib
INSTALL_INCLUDE=/usr/local/include
INSTALL_BIN=/usr/local/bin
```

Dann beginnt die Installation:

```
$ cd /usr/local/src/lamp/gd-1.8.4
$ make
$ make install
```

PHP 4

Hier ist wichtig, dass der Pfad zur Konfigurationsdatei auf /ETC gesetzt ist. Auf die Konfiguration wird noch eingegangen, merken Sie sich den Pfad gut. Sie können nun mit der Installation beginnen:

```
$ cd /usr/local/src/lamp/php-4.0.5
$ ./configure \
  --with-apxs=/usr/local/apache/current/bin/apxs \
  --with-mysql=/usr/local/mysql/current \
  --with-zlib \
  --with-ftp \
  --with-gd \
  --with-jpeg-dir=/usr/local/lib \
  --enable-versioning \
  --enable-track-vars=yes \
  --enable-url-includes \
  --enable-sysvshm=yes \
  --enable-sysvsem=yes \
  --with-config-file-path=/etc
$ make
```

Die PHP.INI wird nach /ETC kopiert.

```
$ make install
$ cp /usr/local/src/lamp/php-4.0.5/php.ini-dist /etc/php.ini
$ mkdir /var/www/htdocs/phpmanual
$ cd /var/www/htdocs/phpmanual
$ tar -xzf /usr/local/src/lamp/tarballs/manual.tar.gz
$ ln -s manual.html index.html
```

Jetzt werden die Rechte zu den Webdokumenten einer Gruppe *wwwuser* zugewiesen:

```
$ chown -R nobody.wwwuser /var/www
$ chmod 700 /usr/local/apache/current/bin/apachectl
```

Die nächsten Schritte brauchen Sie nur machen, wenn dies die erste Installation ist:

```
$ ln -s /usr/local/apache/current/bin/apachectl
  /etc/rc.d/init.d/apachectl
$ cd /etc/rc.d/rc2.d
```

```
$ ln -s ../init.d/apachectl S20apachectl
$ ln -s ../init.d/apachectl K20apachectl
```

Eventuell wird noch ein alter Apache beim Start automatisch aufgerufen – daher bei Bedarf die entsprechenden alten Einträge löschen. Arbeiten Sie auf dem Computer lokal und loggen sich grafisch ein, können Sie die Links für den Runlevel 3 setzen, also den Runlevel mit grafischem Login.

```
$ cd /etc/rc.d/rc3.d
$ ln -s ../init.d/apachectl S20apachectl
$ ln -s ../init.d/apachectl K20apachectl
$ rm S20apache
$ rm K20apache
```

Nun müssen eigentlich nur noch zwei Schritte unternommen werden: Konfiguration der HTTPD.CONF für den Apache und Konfiguration der PHP.INI. Letzteres wird ausführlich in ➔ Abschnitt *Die Konfigurationsdatei php.ini* ab Seite 121 beschrieben.

**Konfiguration
Apache
(httpd.conf)**

Fügen Sie nun der Datei HTTPD.CONF folgende Zeilen hinzu:

```
AddType application/x-httpd-php .php .phtml .php3 .html .htm
```

Empfehlenswert ist auch, wenn neben einer INDEX.HTML auch eine INDEX.PHP als Standardseite konfiguriert wird. Ergänzen Sie die Zeile mit DirectoryIndex folgendermaßen:

```
DirectoryIndex index.html index.php index.htm
```

Jetzt kann der Server gestartet werden:

PHP starten

```
$ /etc/rc.d/init.d/apachectl start
```

Dann ist ein erster Test zu empfehlen:

```
$ lynx localhost
```

2.6 PHP konfigurieren

Um PHP nutzen zu können, sind einige Einstellungen in der Datei PHP.INI notwendig. Lesen Sie diesen Abschnitt aufmerksam, um PHP optimal nutzen zu können.

2.6.1 Die Datei php.ini

Die Einstellungen in PHP.INI stehen Ihnen normalerweise nicht zur Verfügung, wenn Sie PHP auf dem Server eines Providers nutzen. Sie können sich aber mit Hilfe entsprechender Kenntnisse manche Reaktion des Servers erklären.

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Das erste Mal

PHP.INI-DIST

Wenn Sie PHP frisch installieren, finden Sie im Installationsverzeichnis die Datei PHP.INI-DIST. Dies ist die »Distributionsform« – die Rohversion der Initialisierungsdatei. Sie müssen diese Datei in PHP3.INI umbenennen und im Pfad für PHP verfügbar machen. Unter Windows ist es am einfachsten, die Datei ins Windows-Systemverzeichnis zu kopieren. Unter UNIX wird die Datei im aktuellen Arbeitsverzeichnis, also beispielsweise /ETC/PHP, gesucht. Sie können die Position auch festlegen, indem der Pfad in die Umgebungsvariable PHPRC geschrieben wird.



Unter PHP 4 heißt die Konfigurationsdatei nur noch PHP.INI. Bei PHP 3 heißt die Datei PHP3.INI. Darauf wird hier nicht mehr eingegangen.

Verändern Sie unter keinen Umständen die Originaldatei. Wenn Sie PHP mit falschen Einträgen funktionsunfähig machen, können Sie den Originalzustand leicht wiederherstellen, wenn die ursprüngliche Datei noch vorhanden ist.

Aufbau der Datei php.ini

; Kommentar [Sektion]

Der Aufbau entspricht üblichen Konfigurationsdateien. Zeilen, die mit einem Semikolon beginnen, gelten als Kommentare und werden ignoriert. Sektionsabschnitte werden mit eckigen Klammern eingeleitet, beispielsweise [Sektion]. Auch solche Gebilde werden ignoriert.

Groß- und Kleinschreibung

Groß- und Kleinschreibung ist für die Schlüsselwörter relevant. Dies ist ein häufiger Fehler, der beim Eintragen zuvor gelöschter Schlüssel passiert. Achten Sie peinlich genau auf die Schreibweise; an vielen Stellen werden auch mitten im Wort Großbuchstaben zur optischen Auflockerung gesetzt.

Wert "zwei Werte"

Komplexe Werte werden in doppelte Anführungszeichen gesetzt. Ansonsten wertet PHP nur das erste Wort einer Wortkette aus.

Schlüsselwörter: On, Off, Yes, No, True, False

Oft werden Funktionen mit Schlüsselwörtern aktiviert oder deaktiviert. Die Schlüsselwörter sind:

- On, Off
- Yes, No
- TRUE, FALSE, true, false
- 1, 0

Bei den Schlüsselwörtern spielt Groß- und Kleinschreibung keine Rolle, YES und yes sind also identisch.

Die Konfigurationsdatei php.ini

Die folgenden Tabellen zeigen die einzelnen Abschnitte der Datei PHP.INI mit den möglichen Einstellungen. In der Spalte Wert finden Sie ein Beispiel oder den voreingestellten Standardwert.

Option	Wert	Beschreibung
engine	On	Aktiviert die Skriptengine unter Apache
short_open_tag	On	erlaubt kurze Tags: <? anstatt <?php
asp_tags	Off	erlaubt ASP-Tags <% %>
precision	14	Genauigkeit, mit der Gleitkommazahlen verarbeitet werden
y2k_compliance	Off	Jahr 2000-Kompatibilität
output_buffering	Off	Ausgabepufferung; wenn On, ist das Senden von Headern auch nach der Ausgabe von Daten erlaubt.
output_handler		Name einer Funktion, die zur Ausgabe der Seite aufgerufen wird. Schaltet output_buffering implizit ein.
zlib.output_compression	Off	Kompresion des Ausgabestroms mit dem ZIP-Kompressionsverfahren. Kann Off, On oder Puffergröße sein (Standardwert 4KB)
implicit_flush	Off	Wenn On, wird der Puffer nach jedem Block ausgegeben. Entspricht dem Aufruf von flush()
allow_call_time_pass_reference	On	Erlaubt die Übergabe von Funktionsparametern per Referenz. Ist obsolet, da diese Angabe im Funktionskopf erfolgen soll (Operator &).

Tabelle 2.24:
Konfiguration der
Sprachoptionen

Der folgende Abschnitt betrifft Sicherheitsoptionen. Die Standardeinstellungen sind nicht sicher. Provider setzen oft safe_mode auf On, um ihre Server zu schützen. Deshalb gibt es einige Einschränkungen, wenn PHP im Webespace ausgeführt wird. Siehe dazu auch ➡ Abschnitt 2.8 *PHP im Webespace* ab Seite 137.

Option	Wert	Beschreibung
safe_mode	Off	On schaltet den sicheren Modus ein
safe_mode_exec_dir		Erzwingt ein Verzeichnis zur Ausführung

Tabelle 2.25:
Sicherheitsoptionen

Option	Wert	Beschreibung
safe_mode_allowed_env_vars	PHP_	Erzwingt einen Präfix vor Umgebungsvariablen, um den Zugriff aber nicht die Nutzbarkeit einzuschränken. Der Präfix wird mit dieser Option bestimmt.
safe_mode_protected_env_vars	LD_LIBRARY_PATH	Kommaseparierte Liste von Umgebungsvariablen, die nicht verändert werden dürfen.
disable_functions		Kommaseparierte Liste von Funktionsnamen, die nicht genutzt werden dürfen.

PHP erlaubt die Ausgabe von Code mit einer Hervorhebung von syntaktischen Einheiten. Die Farben dieser Ausgabe können Sie im nächsten Abschnitt einstellen. Die Ausgabe der Werte erfolgt in einem ``-Tag.

Tabelle 2.26:
Farben für die
Syntaxhervorhebung

Option	Wert	Beschreibung
highlight.string	#DD0000	Zeichenketten
highlight.comment	#FF8000	Kommentare
highlight.keyword	#007700	Schlüsselwörter
highlight.bg	#FFFFFF	Hintergrund
highlight.default	#0000BB	Alles andere
highlight.html	#000000	HTML

Die folgende Option steuert die Ausgabe eines HTTP-Headers, der auf die Existenz von PHP hinweist. Diese Angabe ist aus Sicherheitsgründen unkritisch.

Tabelle 2.27:
Verschiedenes

Option	Wert	Beschreibung
expose_php	0n	Wenn 0n, fügt PHP einen X-Header hinzu, der die Version anzeigt

Die folgenden Optionen betreffen die Gesamtleistung des Systems. Wenn `safe_mode = 0n`, kann der Wert, der in `max_execution_time` vorgegeben wurde, nicht per Funktion verändert werden.

Tabelle 2.28:
Umgang mit
Ressourcen

Option	Wert	Beschreibung
max_execution_time	30	Maximale Ausführungszeit des Skripts in Sekunden
memory_limit	8M	Maximaler Speicherverbrauch eines Skripts (M = MegaByte)

Der nächste Abschnitt steuert die Fehlerbehandlung. Verwendet werden folgende Konstanten, die Bits eines Bitfelds repräsentieren:

- `E_ALL`. Alle Fehler und Warnungen
- `E_ERROR`. Nur schwerwiegende Fehler
- `E_WARNING`. Nur Warnungen
- `E_PARSE`. Fehler des Compilers
- `E_NOTICE`. Nachrichten
- `E_CORE_ERROR`. Fehler in der Startphase
- `E_CORE_WARNING`. Warnung in der Startphase
- `E_COMPILE_ERROR`. Schwere Compilerfehler
- `E_COMPILE_WARNING`. Compilerwarnungen
- `E_USER_ERROR`. Benutzergenerierte Fehler
- `E_USER_WARNING`. Benutzergenerierte Warnungen
- `E_USER_NOTICE`. Benutzergenerierte Nachrichten

Option	Wert	Beschreibung
<code>error_reporting</code>	<code>E_ALL & ~E_NOTICE</code>	Beispiel für den Einsatz der Konstanten: Alles außer Nachrichten
<code>display_errors</code>	<code>On</code>	Schaltet die Fehlerausgabe ein
<code>display_startup_errors</code>	<code>Off</code>	Fehler beim Startup-Prozess
<code>log_errors</code>	<code>Off</code>	Schaltet die Protokollierung von Fehlern ein
<code>track_errors</code>	<code>Off</code>	Speichert die letzte Fehlermeldung in <code>\$php_errormsg</code>
<code>error_prepend_string</code>	<code><str></code>	Zeichenkette, die vor jedem Fehler angezeigt wird: <code>""</code>
<code>error_append_string</code>	<code><str></code>	Zeichenkette, die nach jedem Fehler ausgegeben wird: <code></code>
<code>error_log</code>	<code><file></code>	Dateiname der Protokolldatei
<code>error_log</code>	<code>syslog</code>	Typ der Systemprotokolldatei (gilt nicht für Windows 9x/Me)
<code>warn_plus_overloading</code>	<code>Off</code>	Warnung einschalten, wenn Zeichenketten mit + verkettet werden

Tabelle 2.29:
Fehlerbehandlung

PHP erleichtert die Programmierung, indem viele Informationen, die zwischen Server und Browser übertragen werden, vorverarbeitet werden. Dieses Verhalten kann in seltenen Fällen problematisch sein. Der folgende Abschnitt erlaubt Modifikationen dieses Verhaltens. Die in früheren Versionen vorhandene Option `track_vars`, die das Scanverhalten aktivierte, gibt es nicht mehr. Dieses Verhalten ist jetzt immer vorhanden.

Tabelle 2.30:
Datenverarbeitung

Option	Wert	Beschreibung
<code>arg_separator</code>	<code>&</code>	Wert, der zum Trennen von Argumenten verwendet wird. Normalerweise ist die Option deaktiviert und es wird ein einfaches <code>&</code> -Zeichen verwendet.
<code>variables_order</code>	<code>"EGPCS"</code>	Reihenfolge, in der externe Daten in Variablen überführt werden (E = Umgebung, G = GET, P = Post, C = Cookies, S = Server)
<code>register_globals</code>	<code>0n</code>	Wenn <code>0n</code> , werden externe, registrierte Variablen global zur Verfügung gestellt, ansonsten nur in den <code>\$HTTP_*_VARS[]</code> -Arrays.
<code>register_argc_argv</code>	<code>0n</code>	Wenn <code>0n</code> , werden Kommandozeilenargumente ausgewertet.
<code>post_max_size</code>	<code>8M</code>	Maximaler Platz für Formulardaten und Dateupload (M = MegaByte)
<code>gpc_order</code>	<code>"GPC"</code>	Obsolet, durch <code>variables_order</code> ersetzt (gilt jedoch für PHP 3).

PHP erlaubt es, Sonderzeichen automatisch mit dem Fluchtzeichen (Escape-Zeichen) Backslash zu versehen, damit diese ihre Wirkung verlieren. Die folgenden Optionen steuern dieses Verhalten.

Tabelle 2.31:
Automatisches
Quoten und
Markieren

Option	Wert	Beschreibung
<code>magic_quotes_gpc</code>	<code>0n</code>	Quoted (markiert Sonderzeichen) in GET-, POST- und Cookie-Daten
<code>magic_quotes_runtime</code>	<code>0ff</code>	Quoted (markiert Sonderzeichen) in allen Zeichenketten
<code>magic_quotes_sybase</code>	<code>0ff</code>	Quoted (markiert Sonderzeichen) im Sybase-Format (mit <code>'</code> -Zeichen anstatt <code>\</code> -Zeichen)

Vor dem Start eines Skripts und danach kann PHP einen speziellen Code ausführen. Tragen Sie hier den Namen der Datei ein, die ausführbaren Skriptcode enthält.

Option	Wert	Beschreibung
auto_prepend_file	<file>	Ausführung vor jedem Skript
auto_append_file	<file>	Ausführung nach jedem Skript

Tabelle 2.32:
Ausführen von
Skripten/HTML vor
und nach jedem
Skript

Bei der Ausgabe von Daten erzeugt PHP einen HTTP-Header, der auf den Inhalt der Seite hinweist. Die folgenden beiden Optionen kennzeichnen die Attribute dieses Headers.

Option	Wert	Beschreibung
default_mimetype	"text/html"	Standard MIME-Type für den HTTP-Header
default_charset	"iso-8859-1"	Setzt den Zeichensatz-Header ein. Soll kein Header gesendet werden, wird die Zeile auskommentiert

Tabelle 2.33:
MIME-Type Header

Bei der Ausführung können Pfade, in denen Skripte liegen, automatisch gefunden werden, wenn die folgenden Optionen konfiguriert sind. Standardmäßig muss der vollständige virtuelle Pfad angegeben werden, wenn Funktionen Datei- oder Pfadangaben erwarten.

Option	Wert	Beschreibung
include_path	<path>	Pfad, in dem nach Dateien gesucht wird. UNIX: "/path1:/path2" Windows: "\\path1;\path2"
doc_root	<path>	Stammpfad (kann leer bleiben)
user_dir	<path>	Aufrufe nach /~username landen in diesem Pfad
extension_dir	./	Verzeichnis mit ladbaren Modulen
enable_dl	On	Schaltet die dl()-Funktion ein

Tabelle 2.34:
Pfade und
Verzeichnisse

PHP erlaubt auf sehr einfache Weise das Hochladen von Dateien. Die folgenden Optionen steuern dieses Verhalten. Es kann jedoch zur Laufzeit mit Funktionen modifiziert werden.

Option	Wert	Beschreibung
file_uploads	On	Erlaubt Upload
upload_tmp_dir	<path>	Temporäres Verzeichnis, in dem die Daten zwischengespeichert werden
upload_max_filesize	2M	Maximale Größe für hochgeladene Dateien (M = MegaByte)

Tabelle 2.35:
Datei-Upload-
Verhalten

Die PHP-Funktion fopen ermöglicht den Zugriff auf Dateien. Spezielle Präfixe vor dem Dateinamen erlauben jedoch den Zugriff auf Dateien

per HTTP oder FTP auf fremden Servern. Mit der folgenden Option lässt sich dieses Verhalten deaktivieren.

Tabelle 2.36:
Wrapper für `fopen()`

Option	Wert	Beschreibung
<code>allow_url_fopen</code>	On	Erlaubt Umleitungen für <code>fopen</code> , beispielsweise <code>http://</code> oder <code>ftp://</code>

PHP erlaubt das Einbinden von Modulen zur Laufzeit. Diese Funktion wird vor allem unter Windows verwendet, wo die Module als DLL vorliegen. Unter Unix ist es eher üblich, Module einzukompilieren. Als SO-Datei vorliegende Unix-Bibliotheken lassen sich aber auch einbinden, wenn sie für PHP übersetzt wurden. Die Liste gibt den derzeitigen Stand der Unterstützung wieder, wie er in der Windows-Distribution von *www.php4win.de* zu erhalten ist. Beachten Sie, dass einige Erweiterungen mit anderen nicht kompatibel sind und PHP 4 deswegen in manchen Konstellationen abstürzen kann. Schalten Sie die benötigten Erweiterungen – und nur diese – deshalb immer schrittweise zu. Aktivieren Sie Module, indem Sie das Semikolon am Anfang entfernen. Beachten Sie, dass unter Windows die Unterstützung für MySQL und ODBC bereits integriert ist und das entsprechende Modul nicht zusätzlich aktiviert werden darf.

Tabelle 2.37:
Dynamische
Erweiterungen
(Auswahl der
wichtigsten DLLs)

Option
<code>extension=php_cpdf.dll</code>
<code>extension=php_cybercash.dll</code>
<code>extension=php_curl.dll</code>
<code>extension=php_db.dll</code>
<code>extension=php_dbase.dll</code>
<code>extension=php_domxml.dll</code>
<code>extension=php_dotnet.dll</code>
<code>extension=php_exif.dll</code>
<code>extension=php_fdf.dll</code>
<code>extension=php_gd.dll</code>
<code>extension=php_gettext.dll</code>
<code>extension=php_ifx.dll</code>
<code>extension=php_imap.dll</code>
<code>extension=php_interbase.dll</code>
<code>extension=php_java.dll</code>
<code>extension=php_ldap.dll</code>

Option
extension=php_mhash.dll
extension=php_mssql65.dll
extension=php_mssql70.dll
extension=php_oci8.dll
extension=php_oracle.dll
extension=php_pdf.dll
extension=php_pgsql.dll
extension=php_sablot.dll
extension=php_swf.dll
extension=php_sybase_ct.dll
extension=php_zlib.dll

PHP besteht aus einer Vielzahl einzelner Module. Die folgenden Abschnitte zeigen Einstellungen für jeweils eines dieser Module.

Abschnitt für Module

Die folgende Option aktiviert oder deaktiviert die Variablen, die Hinweise auf das Systemprotokoll enthalten. Die Werte können auch bei deaktivierter Option mit der Funktion `define_syslog_variables` ermittelt werden.

Option	Wert	Beschreibung
<code>define_syslog_variables</code>	Off	Setzt Systemvariablen, Off verbessert die Systemleistung

Tabelle 2.38:
Modul [Syslog]

Die Mailfunktionen müssen nur unter Windows konfiguriert werden, wo es eine ganze Palette von SMTP-Servern gibt, an die PHP angepasst werden kann. Unter Linux nutzt PHP in jedem Fall `sendmail`, was standardmäßig installiert ist.

Option	Wert	Beschreibung
SMTP	<host>	SMTP-Server (nur für Win32)
<code>sendmail_from</code>	<str>	Standardabsender (nur für Win32)
<code>sendmail_path</code>		Pfad zu <code>sendmail</code> mit Parametern (nur für Unix)

Tabelle 2.39:
Modul [mail
function]

In PHP 3 gab es an dieser Stelle noch Konfigurationen für einen Debugger. Dieser war immer »in Entwicklung« und wurde in PHP 4 offiziell nie realisiert, vermutlich weil Zend hier eine nicht mehr kostenlose Erweiterung in Form einer integrierten Entwicklungsumgebung plant (ZendIDE, siehe ➔ Abschnitt 10.2 *ZendIDE* ab Seite 750).



Die folgenden beiden Optionen steuern das Verhalten des Protokollmoduls, mit dem Fehlerzustände in Protokolldateien geschrieben werden.

Tabelle 2.40:
Modul [Logging]

Option	Wert	Beschreibung
logging.method	db	Speichermethode für Protokolle
logging.directory	<path>	Pfad zu den Protokolldateien

PHP erlaubt die Ausführung von Javacode auf dem Server. Die folgenden Optionen steuern das Verhalten. Die Unterstützung für Java ist wenig ausgereift und schlecht dokumentiert.

Tabelle 2.41:
Java]-Modul für den
Aufruf von Servlets
aus PHP heraus

Option	Wert	Beschreibung
java.class.path	<str>	Pfad zu den Klassenbibliotheken.\PHP_JAVA.JAR
java.home	<str>	Stammverzeichnis:\JDK
java.library	<str>	Die verwendet Java-Bibliothek: \JDK\JRE\BIN\HOTSPOT\JVM.DLL
java.library.path	.\	Pfad zu den externen Bibliotheken

Der folgende Abschnitt betrifft den Umgang mit Datenbankzugriffen. Es folgen dann Optionen für bestimmte Datenbanken.

Tabelle 2.42:
[SQL]-Modul

Option	Wert	Beschreibung
sql.safe_mode	Off	Sicherer Modus für SQL

Das ODBC-Modul betrifft Zugriffe auf Datenbanken, die über eine ODBC-Schnittstelle verfügen. Diese Schnittstelle ist unter Windows weit verbreitet und erlaubt den Zugriff auf Datenquellen wie MS Access oder MS Excel, für die keine nativen Treiber oder Module beiliegen.

Tabelle 2.43:
[ODBC]-Modul

Option	Wert	Beschreibung
odbc.allow_persistent	On	Erlaubt persistente Verbindungen
odbc.check_persistent	On	Prüft persistente Verbindungen vor dem Zugriff
odbc.max_persistent	-1	Maximale Anzahl der persistenten Verbindungen, -1 ist unbegrenzt
uodbc.max_links	-1	Maximale Anzahl der Verbindungen (egal ob persistent oder nicht), -1 ist unbegrenzt
uodbc.defaultlrl	4096	Behandlung von LONG-Feldern, 0 = unverändert

Option	Wert	Beschreibung
uodbc. defaultbinmode	1	Behandlung von Binärdaten, 0 = durchreichen, 1 = keine Veränderung, 2 = Konvertierung in Zeichen

MySQL ist die Standarddatenbank für PHP 4. Entsprechend umfangreich ist die Anzahl der Einstellungsmöglichkeiten. In den meisten Fällen sind die Angaben jedoch unnötig, weil diese Einstellungen bequemer vom Skript aus vorgenommen werden können oder bei aktiviertem Safe-Mode nicht erlaubt sind (betrifft `default_host`, `default_user` und `default_pass`).

Option	Wert	Beschreibung
mysql. .allow_persistent	0n	Erlaubt persistente Verbindungen
mysql.max_persistent	-1	Maximale Anzahl der persistenten Verbindungen, -1 ist unbegrenzt
mysql.max_links	-1	Maximale Anzahl der Verbindungen (egal ob persistent oder nicht), -1 ist unbegrenzt
mysql.default_port	<port>	Standardport des MySQL-Servers
mysql.default_socket	<sock>	Socket für den MySQL-Server (bleibt normalerweise leer)
mysql.default_host	<host>	Name des MySQL-Servers. Kann im sicheren Modus nicht verändert werden.
mysql.default_user	<user>	Standardbenutzer. Kann im sicheren Modus nicht verändert werden.
mysql. .default_password	<pass>	Standardkennwort. Kann im sicheren Modus nicht verändert werden.

Tabelle 2.44:
Modul [MySQL]

Auch die Open-Source-Datenbank mSQL ist sehr beliebt. Die folgenden Optionen betreffen dieses Modul, wenn es installiert wurde.

Option	Wert	Beschreibung
msql.allow_persistent	0n	Erlaubt persistente Verbindungen
msql.max_persistent	-1	Maximale Anzahl der persistenten Verbindungen, -1 ist unbegrenzt
msql.max_links	-1	Maximale Anzahl der Verbindungen (egal ob persistent oder nicht), -1 ist unbegrenzt

Tabelle 2.45:
Modul [mSQL]

Die dritte »große« Open-Source-Datenbank ist PostgreSQL. Auch dafür gibt es ein Modul in PHP, das diese Datenbank direkt unterstützt.

Tabelle 2.46:
Modul [PostgreSQL]

Option	Wert	Beschreibung
pgsql.allow_persistent	0n	Erlaubt persistente Verbindungen
pgsql.max_persistent	-1	Maximale Anzahl der persistenten Verbindungen, -1 ist unbegrenzt
pgsql.max_links	-1	Maximale Anzahl der Verbindungen (egal ob persistent oder nicht), -1 ist unbegrenzt

SyBase ist eine kommerzielle Datenbank, die für viele Betriebssysteme verfügbar ist.

Tabelle 2.47:
Module [Sybase] und
[SybaseCT]

Option	Wert	Beschreibung
sybase.allow_persistent	0n	Erlaubt persistente Verbindungen
sybase.max_persistent	-1	Maximale Anzahl der persistenten Verbindungen, -1 ist unbegrenzt
sybase.max_links	-1	Maximale Anzahl der Verbindungen (egal ob persistent oder nicht), -1 ist unbegrenzt
sybase.interface_file	<str>	Schnittstellendatei. "/usr/sybase/interfaces"
sybase.min_error_severity	10	Minimales Fehlerniveau zur Fehleranzeige
Sybase.min_message_severity	10	Minimales Nachrichtenniveau zur Nachrichtenanzeige
sybase.compatibility_mode	Off	Kompatibilitätsmodus für PHP 3
sybct.allow_persistent	0n	Erlaubt persistente Verbindungen
sybct.max_persistent	-1	Maximale Anzahl der persistenten Verbindungen, -1 ist unbegrenzt
sybct.max_links	-1	Maximale Anzahl der Verbindungen (egal ob persistent oder nicht), -1 ist unbegrenzt
sybct.min_server_severity	10	Minimales Nachrichtenniveau zur Nachrichtenanzeige

Option	Wert	Beschreibung
sybct.min_client_severity	10	Minimales Nachrichtenniveau zur Nachrichtenanzeige

Normalerweise werden Zahlenwerte im Computer in binärer Form gespeichert. Dies führt jedoch in seltenen Fällen zu Rechenfehlern, weil der Wertebereich anders als bei dezimalen Zahlen ausgebildet ist. Werden Berechnungen mit hoher Genauigkeit benötigt, hilft das BCMATH-Modul, dass intern mit dezimalen Zahlen rechnet. Die Anzahl der gespeicherten Dezimalstellen können Sie mit der folgenden Option angeben.

Option	Wert	Beschreibung
bcmath.scale	0	Anzahl der Dezimalstellen bei bc-Funktionen, 0 = Keine Vorgabe

Tabelle 2.48:
[bcmath]-Modul

Um zu erkennen, welche Funktionen ein Browser hat, wird eine Kennungsdatei verwendet, die für alle bekannten Browser entsprechende Angaben enthält. Dies sagt jedoch nichts darüber aus, ob der Benutzer diese Funktionen auch aktiviert hat. Geben Sie mit der folgenden Option den Pfad zu einer aktuellen Version dieser Datei an.

Option	Wert	Beschreibung
browscap	<file>	Pfad zur Browserinformationsdatei

Tabelle 2.49:
Modul [browscap]

Informix ist eine weitere kommerzielle Datenbank, die vor allem in der Unix-Welt verbreitet ist.

Option	Wert	Beschreibung
ifx.default_host	<host>	Standardhost. Kann im sicheren Modus nicht verändert werden.
ifx.default_user	<user>	Standardbenutzer. Kann im sicheren Modus nicht verändert werden.
ifx.default_password	<pass>	Standardkennwort. Kann im sicheren Modus nicht verändert werden.
ifx.allow_persistent	On	Erlaubt persistente Verbindungen
ifx.max_persistent	-1	Maximale Anzahl der persistenten Verbindungen, -1 ist unbegrenzt
ifx.max_links	-1	Maximale Anzahl der Verbindungen (egal ob persistent oder nicht), -1 ist unbegrenzt

Tabelle 2.50:
[Informix]-Modul

Option	Wert	Beschreibung
ifx.textasvarchar	0	Wenn 1, werden BLOB-Felder direkt zurückgeliefert, anstatt nur eine ID darauf (Textfelder)
ifx.byteasvarchar	0	Wenn 1, werden BLOB-Felder direkt zurückgeliefert, anstatt nur eine ID darauf (Bytefelder)
ifx.charasvarchar	0	Anhängende Leerzeichen werden beim Lesen von Spalten fester Breite entfernt
ifx.blobinfile	0	Wenn 1, werden BLOB-Felder in einer Datei abgelegt
ifx.nullformat	0	Wenn 0, werden NULL-Werte als leere Zeichenkette zurückgegeben, bei 1 als 'NULL'

Das Session-Modul steuert die Verfolgung von Nutzern über mehrere Seiten hinweg. Dies ist notwendig, weil HTTP selbst statuslos arbeitet und die Verbindung nach der Auslieferung jeder Seite beendet. Die folgenden Optionen steuern das Verhalten von PHP in Bezug auf Sitzungen.

Tabelle 2.51:
[Session]-Modul

Option	Wert	Beschreibung
session.save_handler	files	Art der Sessionspeicherung (files = in Dateien). Kann auch user sein (eigene Funktionen erforderlich).
session.save_path	<path>	Pfad, wo die Sessionsteuerung Daten speichert (meist /tmp)
session.use_cookies	1	Erlaubt Cookies
session.name	PHPSESSID	Name der Session, wird als Cookie-name oder GET-Parameter verwendet
session.auto_start	0	Startet Session immer auch ohne session_start-Funktion
session.cookie_lifetime	0	Lebensdauer des Cookies in Sekunden, 0 = Sessioncookie
session.cookie_path	/	Pfad für Cookie
session.cookie_domain	<dom>	Domäne für Cookie
session.serialize_handler	php	Art der Serialisierung der Daten (php = interne Funktion)

Option	Wert	Beschreibung
session.gc_probability	1	1 startet den Aufräumprozess bei jeder Initialisierung einer Session
session.gc_maxlifetime	1440	Zeit, nach der temporäre Daten gelöscht werden
session.referer_check	0	Prüft die Herkunft des Skripts
session.entropy_length	0	Anzahl Bytes, die maximal gelesen werden, 0 = unbegrenzt
session.entropy_file	<name>	Datei, aus der Session-IDs generiert werden
session.entropy_length	16	Länge der Session-ID (32, wenn auskommentiert = Standard)
session.entropy_file	/dev/urandom	Quelle für den Zufallsgenerator, Standard = auskommentiert (deaktiviert)
session.cache_limiter	no cache	Lebensdauer des Caches im Browser (nocache, private, public)
session.cache_expire	180	Lebensdauer des Dokuments
session.use_trans_sid	1	Erlaubt transiente Übergabe der Session-ID, wenn mit --enable-trans-sid compliert wurde
url_rewriter.tags	<str>	Liste der Tags, die erweitert werden, um die Session-ID automatisch anzuhängen: "a=href,area=href,frame=src,input=src,form=fakeentry"

Unter Windows ist der MS SQL Server die leistungsfähigste Datenbank. PHP erlaubt die direkte Ansteuerung eines MS SQL Servers. Die nachfolgende Tabelle zeigt die verfügbaren Standardoptionen.

Option	Wert	Beschreibung
mssql.allow_persistent	On	Erlaubt persistente Verbindungen
mssql.max_persistent	-1	Prüft persistente Verbindungen vor dem Zugriff
mssql.max_links	-1	Maximale Anzahl der persistenten Verbindungen, -1 ist unbegrenzt
mssql.min_error_severity	10	Minimales Fehlerniveau

Tabelle 2.52:
[MSSQL]-Modul

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Option	Wert	Beschreibung
mssql.min_message_severity	10	Minimales Meldungsniveau
mssql.compatibility_mode	Off	Kompatibilitätsmodus für PHP 3
mssql.textlimit	4096	Begrenzung eines Textfelds, 0 bis 2 147 483 647; 4 096 = Standard
mssql.textsize	4096	Größe eines Textfelds, 0 bis 2 147 483 647; 4 096 = Standard
mssql.batchsize	0	Anzahl der Elemente einer Stapeldatei, 0 = unbegrenzt

Zum Zweck der Fehlersuche kann eine Fehlerverfolgungsfunktion genutzt werden. Deren Optionen werden nachfolgend gezeigt.

Tabelle 2.53:
[Assertion]-Modul

Option	Wert	Beschreibung
assert.active	On	Aktiviert die Funktion assert()
assert.warning	On	Gibt eine Warnung bei jeder misslungenen Prüfung aus
assert.bail	Off	Lässt Alternative standardmäßig nicht zu
assert.callback	0	Funktion, die bei misslungener Prüfung aufgerufen wird
assert.quiet_eval	0	Setzt error_reporting() für die Zeit der Prüfung außer Kraft, wenn 1

Eine weitere kommerzielle Datenbank, die PHP unterstützt, ist Ingres. Die folgende Tabelle zeigt die verfügbaren Standardoptionen.

Tabelle 2.54:
[Ingres II]-Modul

Option	Wert	Beschreibung
ingres.allow_persistent	On	Erlaubt persistente Verbindungen
ingres.max_persistent	-1	Prüft persistente Verbindungen vor dem Zugriff
ingres.max_links	-1	Maximale Anzahl der persistenten Verbindungen, -1 ist unbegrenzt
ingres.default_database	<str>	Standarddatenbank (<[node_id::]dbname[/srv_class]>)
ingres.default_user	<user>	Standardbenutzer

Option	Wert	Beschreibung
ingres .default_password	<pass>	Standardkennwort

Für Zahlungsfunktionen wurde das Payflow-Modul von Verisign integriert. Dieses ist jedoch in Europa noch wenig gebräuchlich und bei Providern nur selten vorhanden.

Option	Wert	Beschreibung
pfpro.defaultthost	"test.s ignio.c om"	Signio-Server
pfpro.defaultport	443	Standardport
pfpro .defaulttimeout	30	Zeitüberschreitung in Sekunden
pfpro.proxyaddress	<ip>	Proxyadresse
pfpro.proxyport	<port>	Proxyport
pfpro.proxylogin	<name>	Proxybenutzer
pfpro .proxypassword	<pass>	Proxykennwort

Tabelle 2.55:
Modul [Verisign
Payflow Pro]

Beim Aufbau einer Verbindung per TCP/IP wird eine Kombination aus IP-Adresse und Port benutzt – ein sogenanntes Socket. Dieses kann vom Betriebssystem oder von PHP verwendet werden, was die folgende Option steuert.

Option	Wert	Beschreibung
sockets.use_system _read	0n	Wenn 0n, wird der Systemaufruf <i>read</i> anstatt des mit PHP gelieferten genutzt

Tabelle 2.56:
[Sockets]-Modul

2.7 Sicherheit

Die Absicherung eines Webservers sollte immer mit großer Aufmerksamkeit erfolgen. Es gibt viele Gründe, die zu einem Angriff führen können.

2.7.1 Sicherheitsprobleme

PHP ist eine sehr leistungsstarke Sprache, die viele Befehle und Funktionen enthält, mit denen direkt auf das Betriebssystem eingewirkt werden kann. Auch wenn Sie die Kontrolle über die Skripte behalten – schon ein einfacher Tippfehler kann problematisch für die Sicherheit des Webservers sein. PHP kann Dateien lesen und schreiben, löschen

und verändern. PHP wird als CGI-Programm oder als Modul im Apache-Webserver installiert. Auch aus dieser Position heraus kann mit PHP-Skripten Schaden angerichtet werden.

PHP selbst ist mit weitreichenden Freiheiten ausgestattet, jedoch keineswegs in dem Umfang wie C oder auch Perl. Die Sicherheitseinstellungen sind vielfältig modifizierbar.

2.7.2 Angriffsszenarien

Angriffe können jeden treffen

Angriffe auf Webserver sind häufig und treffen – früher oder später – jeden Webserver. Inzwischen scheinen nicht nur einige wenige engagierte Hacker Jagd auf bekannte Sites zu machen. Mit den im Internet grassierenden Hacker-Tools entwickelt sich eine Art Volkssport. Dabei geht es um den Lerneffekt und Spaß an der Sache, keine Site ist unbedeutend und unwichtig genug, um nicht doch zum Opfer werden zu können. Auch wenn keine wichtigen Daten vorhanden sind, kann ein Angriff zu Schäden führen und kostet zumindest die wertvolle Zeit des Systemadministrators.

PHP als CGI-Programm

Parameterattacke

Die Installation als CGI-Programm ist der wichtigste Sicherheitsschritt. Dem steht die Leistung entgegen – CGI ist als universelle Schnittstelle ausgesprochen langsam. Zum Hintergrund ist es wichtig zu wissen, wie die PHP-Binärdateien arbeiten. Generell wird jeder Parameter als Kommandozeilenparameter interpretiert. Dies entspricht dem Aufruf des PHP-Interpreters am Prompt mit PHP. Nun könnte ein Angreifer in seinem Browser die URL so manipulieren:

```
http://www.meinserver.de/cgi-bin/php?etc/password
```

PHP würde den QueryString (die Zeichen nach dem ?-Zeichen) als Kommandozeilenparameter interpretieren und die Kennwortdatei übertragen. Dies ist sicher nicht erwünscht. Im CGI-Modus unterdrückt PHP diese Art der Interpretation selbst und kopiert den QueryString in eine entsprechende Variable. In Kapitel 4 wird darauf sehr ausführlich eingegangen.

Pfadattacke

Das nächste Problem stellen manipulierte Pfade dar. Normalerweise kann ein Nutzer aus dem Internet nicht auf Pfade des Webserver zugreifen, die nicht explizit freigegeben wurden, entweder unter HTDOCS (Apache) oder WWWROOT (IIS). Ist PHP im /CGI-BIN-Verzeichnis erreichbar, sind Pfadangaben wie die folgende erlaubt:

```
http://www.meinserver.de/cgi-bin/php/docs/bilanz.doc
```

Der an den Aufruf des Programms PHP angehängte Pfad, hier /DOCS/BILANZ.DOC wird als Zugriff auf ein Dokument interpretiert.

Das Dokument wird geholt und an den Browser gesendet. Normalerweise ist der Webserver in der Lage, die Zugriffsrechte zu überprüfen. Im gezeigten Fall werden Sie vielleicht das Dokument nur einer geschlossenen Benutzergruppe zugänglich machen wollen. Dann richten Sie ein virtuelles Verzeichnis ein und schließen den anonymen Webnutzer (*nobody* im Apache, *IUSR_Machine* im IIS) davon aus. So erfolgt der Aufruf:

```
http://www.meinserver.de/docs/bilanz.doc
```

Beim Abruf prüft der Webserver die Rechte und verlangt Name und Kennwort oder lehnt die Übertragung sofort ab. Wird aber der physische Pfad (wenn er denn bekannt ist) verwendet, prüft der Webserver nur die Sicherheit der Pfadangabe vor »php«:

```
http://www.meinserver.de/cgi-bin/php/docs/bilanz.doc
```

Normalerweise wird der Interpreter aber allen zugänglich sein, das Dokument wird übertragen. Mit den Schlüsselwörtern `doc_dir` und `user_dir` kann PHP Voreinstellungen der Stammpfade einrichten. Damit kann das Verhalten unterdrückt werden, nur die hier angegebenen Pfade werden überhaupt akzeptiert. Die Einstellungen werden im entsprechenden Abschnitt der Datei `PHP.INI` vorgenommen und wurden bereits in ➤ Abschnitt 2.6 *PHP konfigurieren* ab Seite 119 beschrieben.

`doc_dir`
`user_dir`

PHP außerhalb der Webserver-Umgebung

Die sicherste Methode ist die Platzierung der ausführbaren Dateien außerhalb der Webserver-Umgebung, beispielsweise in `/usr/local/bin` (Unix) oder im Installationsverzeichnis `C:\PHP` (Windows). Sie können mit Hilfe der Apache-Konfigurationsdatei `HTTPD.CONF` die Verknüpfungen an diesem Ort herstellen oder im IIS die entsprechende Verknüpfung vornehmen.

In der Unix-Umgebung müssen Sie allerdings jede Datei mit dem Shell-Kommando `#!` beginnen und den Pfad zu PHP angeben. Perl-Programmierer kennen dieses Verfahren sicher:

```
#!/usr/local/bin/php
```

Darüber hinaus müssen die Dateien Ausführrechte haben, nicht nur Skriptrechte.

2.8 PHP im Webospace

Der häufigste Fall dürfte die Nutzung von PHP im Webospace sein. Inzwischen bieten fast alle Provider PHP an, in den meisten Fällen auch in der neuesten Version PHP 4.

2.8.1 Vorgegebene Daten

Normalerweise sind Einstellungsarbeiten nicht notwendig. Sie können auf die PHP.INI zwangsläufig nicht zugreifen, denn dies würde für den Provider eine riesige Sicherheitslücke darstellen.

Wenn Sie Webspace angemietet haben, wird Ihnen normalerweise folgendes mitgeteilt:

- *Hostname und IP-Adresse*

Der Hostname wird die von Ihnen gewählte Domain sein.

- *FTP-Server*

Meist ist dies der gleiche Name wie der Host, manchmal wird der Vorsatz *www* auch gegen *ftp* ausgetauscht.

- *Login-Name*

Mit diesem Namen können Sie per FTP auf den Web-space zugreifen.

- *Kennwort*

Das Kennwort schützt den Zugang entsprechend.

Wenn Sie zusätzlich eine Unterstützung für MySQL haben, wird auch dafür Name und Kennwort, gegebenenfalls auch ein weiterer Hostname zugeteilt. Teilweise sind diese Angaben mit den oben bereits genannten identisch.

Was unterstützt mein Provider?

Trotz aller Informationen auf den Webseiten halten sich die meisten Provider sehr bedeckt, was konkrete Informationen über den genutzten Webspace betrifft. Es ist eine gute Idee, auch ohne entsprechendes Wissen über PHP den frisch angemieteten Webspace zu testen. Dazu geben Sie in einem Texteditor folgende Zeilen ein:

```
<?php
phpinfo();
?>
```

Speichern Sie dieses Skript unter dem Namen INFO.PHP ab und laden Sie es dann per FTP in das Rootverzeichnis Ihres Webspace. Dann öffnen Sie Ihren Browser und geben folgende Adresse ein:

```
http://www.ihrwebspace.de/info.php
```

Es sollte nun eine längere Ausgabe erfolgen, die etwa den folgenden Abbildungen entspricht. Diese »Serverkonfiguration« ist nun für alle weiteren Projekte die Basis. Die Abbildungen in ➡ Abschnitt 2.8.2 zeigen die verfügbaren PHP-Erweiterungen. Wichtig sind folgende Punkte:

- *Versionsnummer*

Das Buch geht weitestgehend von Version 4.0.6 aus, die aktuellste Version finden Sie unter <http://www.php.net>. Viele Beispiele laufen auch mit einer niedrigeren Version, auch mit PHP 3. Als unterstes Niveau sollte aber 3.0.12 verfügbar sein.

- *Datenbankunterstützung*

Im Beispiel wird neben MySQL auch PostgreSQL angeboten.

- *Grafikbibliothek*

Die dynamische Erzeugung von Bildern wird durch die GD-Bibliothek mit FreeType erledigt.

- *Perl Regular Expressions*

Wenn Sie professionell mit regulären Ausdrücken umgehen möchten, sind die Perl-Erweiterungen sinnvoll.

2.8.2 Überprüfen der Konfiguration

Mit Hilfe des Befehls `phpinfo` können Sie die Konfiguration überprüfen. Das ist sinnvoll, um eigene Änderungen bestätigt zu bekommen, aber auch um im Webespace die Einstellungen des Providers herauszubekommen.

Die folgenden Abbildungen zeigen eine typische Konfiguration mit allen Werten, die `phpinfo` erzeugt.



PHP Version 4.0.7-dev	
	
System	Windows NT 5.0 build 2195
Build Date	Jun 11 2001
Server API	CGI
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINNT\php.ini
ZEND_DEBUG	disabled
Thread Safety	enabled

Abbildung 2.4:
Konfiguration eines
PHP-Servers unter
Windows 2000 (als
CGI-Version)

Die Anzeige der Daten kann sich je nach Version unterscheiden. Die hier gezeigten Bildschirmfotos sind nur als Orientierung zu verstehen.

Abbildung 2.5:
Ausgabe auf einem
Linux-Server,
ebenfalls als CGI-
Version

PHP Version 4.0.6 	
System	Linux infong 2.2.20 #2 SMP Wed Jun 20 14:48:31 CEST 2001 i686 unleashed
Build Date	Jun 26 2001
Configure Command	'./configure' '--with-mysql' '--with-mysql=/usr/local/mysql' '--with-pgsql=/usr/lib/pgsql' '--with-zlib' '--enable-debug=no' '--enable-safe-mode=no' '--enable-discard-path=no' '--with-gd' '--with-png-dir=/usr/local/lib' '--enable-track-vars' '--with-db' '--with-gdbm' '--enable-force-cgi-redirect' '--with-ttf=/usr/src/kundenserver/freetype-1.2/' '--enable-ftp' '--with-mcrypt' '--enable-dbase' '--enable-memory-limit' '--enable-calendar' '--enable-wddx' '--enable-trans-sid' '--with-jpeg-dir=/usr/src/kundenserver/jpeg-6b' '--enable-bcmath' '--enable-gd-imgstrttf' '--enable-shmop' '--enable-mhash' '--with-mhash=/usr/src/kundenserver/mhash-0.8.9/' '--with-openssl'
Server API	CGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/lib/php.ini
ZEND_DEBUG	disabled
Thread Safety	disabled

Die auf der folgenden Seite gezeigten Konfigurationseinstellungen entsprechen den Angaben in der PHP.INI. Wenn Sie PHP lokal oder auf einem eigenen Server verwenden, können Sie diese Werte beeinflussen. Falls Sie PHP im Webpace Ihres Providers verwenden, müssen Sie wahrscheinlich die Einstellungen als gegeben und unveränderlich ansehen. Sie können Ihre Skripte dann aber darauf abstimmen. In vielen Fällen stehen Funktionen zur Verfügung, mit denen sich die Standardeinstellungen umgehen lassen.

Abbildung 2.6:
Standard-
einstellungen

standard		
Regex Library	Bundled library enabled	
Dynamic Library Support	enabled	
Internal Sendmail Support for Windows 4	enabled	
Directive	Local Value	Master Value
assert.active	1	1
assert.bail	0	0
assert.callback	no value	no value
assert.quiet_eval	0	0
assert.warning	1	1
safe_mode_allowed_env_vars	PHP_	PHP_
safe_mode_protected_env_vars	LD_LIBRARY_PATH	LD_LIBRARY_PATH

com		
Directive	Local Value	Master Value
allow_dcom	On	On
typelib_file	no value	no value

pcre	
PCRE (Perl Compatible Regular Expressions) Support	enabled
PCRE Library Version	3.1 09-Feb-2000

Abbildung 2.7:
Weitere Module:
COM steht nur
unter Windows zur
Verfügung

odbc	
ODBC Support	enabled
Active Persistent Links	0
Active Links	0
ODBC library	Win32

Directive	Local Value	Master Value
odbc.allow_persistent	On	On
odbc.check_persistent	On	On
odbc.default_db	no value	no value
odbc.default_pw		
odbc.default_user	no value	no value
odbc.defaultbinmode	return as is	return as is
odbc.defaultlrl	return up to 4096 bytes	return up to 4096 bytes
odbc.max_links	Unlimited	Unlimited
odbc.max_persistent	Unlimited	Unlimited

Abbildung 2.8:
ODBC-Unter-
stützung wird für
den Zugriff auf
Access benötigt

Die in der folgenden Abbildung gezeigte Liste von Einstellungen umfasst fast alle Parameter, die über die Datei PHP.INI eingestellt werden können. Die Angabe ist vor allem im Webespace von Bedeutung, wo sich die meisten Werte zwar nicht beeinflussen lassen, Sie aber darüber Bescheid wissen sollten, wie PHP vom Provider konfiguriert wurde. In vielen Fällen lassen sich Skripte so programmieren, dass sie auf besondere Bedingungen nicht angewiesen sind – es ist eine gute Idee, die Einstellungen als »gegeben« zu nehmen und Probleme programmtechnisch zu lösen. Wenn Sie allzu intensive Veränderungen an der Konfiguration vornehmen, wird Ihr Skript möglicherweise nicht auf allen Servern laufen.

Abbildung 2.9:
Die PHP-Konfiguration. Die meisten Werte können in der `php.ini` eingestellt werden.

Configuration		
PHP Core		
Directive	Local Value	Master Value
allow_call_time_pass_reference	On	On
arg_separator	&	&
asp_tags	Off	Off
auto_append_file	no value	no value
auto_prepend_file	no value	no value
browscap	no value	no value
default_charset	no value	no value
default_mimetype	text/html	text/html
define_syslog_variables	Off	Off
disable_functions	no value	no value
display_errors	On	On
doc_root	no value	no value
enable_dl	On	On
error_append_string	Off	Off
error_log	no value	no value
error_prepend_string	Off	Off
error_reporting	2039	2039
expose_php	On	On
extension_dir	./	./
gpc_order	GPC	GPC
highlight.bg	#FFFFFF	#FFFFFF
highlight.comment	#FF8000	#FF8000
highlight.default	#0000BB	#0000BB
highlight.html	#000000	#000000
highlight.keyword	#007700	#007700
highlight.string	#DD0000	#DD0000
ignore_user_abort	Off	Off
implicit_flush	Off	Off
include_path	no value	no value
log_errors	Off	Off
magic_quotes_gpc	On	On
magic_quotes_runtime	Off	Off
magic_quotes_sybase	Off	Off
max_execution_time	60	60
open_basedir	no value	no value
output_buffering	Off	Off
precision	14	14
register_argc_argv	On	On
register_globals	On	On
safe_mode	Off	Off
safe_mode_exec_dir	no value	no value
sendmail_from	me@localhost.com	me@localhost.com
sendmail_path	no value	no value
short_open_tag	On	On
SMTP	localhost	localhost
sql.safe_mode	Off	Off
track_errors	Off	Off
track_vars	On	On
upload_max_filesize	2097152	2097152
upload_tmp_dir	no value	no value
user_dir	no value	no value
variables_order	EGPCS	EGPCS
y2k_compliance	Off	Off

session		
Session Support		enabled
Directive	Local Value	Master Value
session.auto_start	Off	Off
session.cache_expire	180	180
session.cache_limiter	nocache	nocache
session.cookie_domain	no value	no value
session.cookie_lifetime	0	0
session.cookie_path	/	/
session.entropy_file	no value	no value
session.entropy_length	0	0
session.gc_maxlifetime	1440	1440
session.gc_probability	1	1
session.name	PHPSESSID	PHPSESSID
session.referer_check	no value	no value
session.save_handler	files	files
session.save_path	/tmp	/tmp
session.serialize_handler	php	php
session.use_cookies	On	On

Abbildung 2.10:
Die Variablen des
Session-Management

xml	
XML Support	active

Abbildung 2.11:
XML wird
unterstützt

mysql		
MySQL Support		enabled
Active Persistent Links	0	
Active Links	0	
Client API version	3.23.10-alpha	
Directive	Local Value	Master Value
mysql.allow_persistent	On	On
mysql.default_host	no value	no value
mysql.default_password	no value	no value
mysql.default_port	no value	no value
mysql.default_socket	no value	no value
mysql.default_user	no value	no value
mysql.max_links	Unlimited	Unlimited
mysql.max_persistent	Unlimited	Unlimited

Abbildung 2.12:
Einstellungen für
MySQL

Abbildung 2.13:
Unterstützung für
weitere Module.
Diese Angabe ist
davon abhängig, was
mit kompiliert wurde.

Calendar	
Calendar support enabled	
Additional Modules	
wddx	
mhash	
mcrypt	
ftp	
dbase	

Es können weitere Informationen, beispielsweise zu Zusatzprodukten von Zend, wie der Zend Optimizer folgen. Den letzten Teil bilden dann Übersichten über Systemvariablen und Webservervariablen, die Abhängig von der Installation und aktuellen Situation sind.

3



Erste Schritte mit PHP

Dieses Kapitel zeigt die Grundfunktionen in PHP anhand vieler praktischer Beispiele. Anfänger finden die Grundlagen, um schnell erste Erfolge erzielen zu können. Umsteiger erhalten einen Überblick über die Leistungsfähigkeit der Sprache.

Aufbau der Skripte (Seite 147)

Eine kompakte Einführung in PHP (Seite 151)

Programmieren mit PHP (Seite 230)

Objektorientierte Programmierung (Seite 254)

Fehlerbehandlung (Seite 271)

Hilfsfunktionen (Seite 294)

3.1 Aufbau der Skripte

Dieser Abschnitt zeigt, wie Skripte aufgebaut sind, wo und wie Sie PHP schreiben und welche Zusammenhänge es zwischen PHP, HTML und JavaScript gibt.

3.1.1 Wo wird programmiert?

PHP ist eine Skriptsprache, deren Elemente direkt in HTML-Seiten eingebaut werden. Durch die Dateierweiterung `.PHP`, `.PHP3`, `.PHP4` oder `.PHTML` wird dann daraus eine PHP-Skriptdatei. Um PHP-Skripte ablaufen lassen zu können, muss ein entsprechend eingerichteter Server vorhanden sein. Wie das für ein Entwicklungssystem installiert wird, wurde bereits in Kapitel 2 behandelt.

Schreiben können Sie PHP mit allen gängigen Editoren. Wenn Sie an speziellen Features wie Syntaxhervorhebung oder automatische Vervollständigung interessiert sind, sieht das Angebot noch etwas dürftig aus. Im Gegensatz zu JavaScript oder VBScript ist PHP noch relativ wenig verbreitet. Die Zunahme an PHP-Sites ist aber augenfällig und die Softwarehersteller werden ihre Produkte sehr schnell umstellen. Einige heute verfügbare Editoren werden in Kapitel 10 *Entwicklungsumgebungen* vorgestellt.

Sie können PHP am einfachsten über den Webserver ausführen. **PHP ausführen**
Grundsätzlich lassen sich zwar Skripte auch über die Shell starten, dies macht jedoch wenig Sinn, denn die HTML-Ausgaben werden fortlaufend am Bildschirm ausgegeben.

Beachten Sie, dass der Webserver nicht mit einem lokalen Pfad angesprochen werden kann. Sie müssen eine URL eingeben. Den lokalen Rechner erreichen Sie normalerweise als »localhost«.

3.1.2 PHP und HTML

PHP wird im Gegensatz zu Perl nicht als eigenständiges CGI-Programm ausgeführt, sondern direkt im HTML-Code untergebracht. Dazu muss natürlich eine Möglichkeit geschaffen werden, wie der PHP-Prozessor die für ihn bestimmten Seiten erkennt und interpretiert.

Dateierweiterung

Der erste Schritt besteht in der Benennung einer HTML-Seite mit einer anderen Dateierweiterung. Die kennen bisher Erweiterungen wie `*.HTM` oder `*.HTML`. Sie können jede HTML-Datei einfach mit der Erweiterung `*.PHP3` (PHP 3), `*.PHP4` (PHP 4) oder `*.PHP` (PHP 3 und

PHP 4) versehen. Jetzt wird bei einem Aufruf dieser Datei der Webserver die Seite nicht sofort zum Browser senden, sondern zuerst an den PHP-Prozessor übergeben. Der PHP-Prozessor arbeitet im Apache als Modul oder als externes CGI-Programm. PHP 3 unter Windows kann nur über CGI angesprochen werden. PHP 4 gibt es auch als ISAPI-Anwendung.

Der PHP-Prozessor arbeitet nun die Datei von oben nach unten ab. Reine HTML-Seiten wird er unverändert wieder zurückgeben. Sie werden davon nichts bemerken, abgesehen von der höheren Belastung Ihres Servers und, bei langsamen Maschinen, einer verringerten Ausgabegeschwindigkeit.

Finden sich in der Datei PHP-Anweisungen, werden diese erkannt, ausgewertet und ausgeführt.

Die Erweiterung im Buch

Im Buch und auf der Website wird konsequent die Erweiterung PHP4 verwendet. Wenn Sie die Skripte auf einem anders konfigurierten Webserver verwenden, müssen Sie die Endungen und alle Aufrufe von Dateien in den Skripten ändern.

PHP einbinden

Es gibt insgesamt vier Möglichkeiten, PHP in der Seite unterzubringen. Damit ist PHP die universellste aller Skriptsprachen.

SGML-Style

Die erste ist der sogenannte SGML-Style:

```
<?
    // hier steht der Code
?>
```

XML-Style

Da sich XML als erweiterter Standard etablieren wird, können Sie die dort übliche Notation auch in PHP nutzen:

```
<?PHP
    // hier steht der Code
?>
```

Die Skripte im Buch verwenden einheitlich die kleine Schreibweise:

```
<?php
    // hier steht der Code
?>
```

ASP-Style

ASP-Programmierer sind typische Umstiegskandidaten für PHP. Damit es nicht so schwierig wird, können Sie die bei ASP übliche Symbolik ebenso nutzen:

```
<%
    // hier steht der Code
%>
```

Um ASP-Style-Tags nutzen zu können, muss die Option `asp_tags` in `php.ini` der `PHP.INI` auf `On` stehen.

Beachten Sie, dass öffentliche Webserver bei Providern oft auf Linux basieren und dort die Nutzung der ASP-Styles nicht erlaubt ist, und Sie dies ohne Zugriff auf die `PHP.INI` auch nicht ändern können. Diese Option ist deshalb *nicht* empfehlenswert!



Wenn Sie bereits mit JavaScript gearbeitet haben, wird Ihnen die folgende Version vielleicht besser gefallen:

```
<script language="php">
  // hier steht der Code
</script>
```

Wenn Sie zukunftsorientiert programmieren wollen, sollten Sie den XML-Stil verwenden: `<?php ... ?>`. XML wird sich in den nächsten Jahren zu einer umfassenden Metasprache entwickeln, da sich spezielle Sprachen über Beschreibungsdokumente einfach definieren lassen.



Alle Skripte in diesem Buch und auf der Website zum Buch verwenden die Markierung mit `<?php ... ?>`.

Notationshinweise

Am Anfang ist die Versuchung groß, unübersichtliche Skripte zu bauen. Denn Leerzeichen, Zeilenumbrüche und Tabulatoren spielen keine Rolle. Sie können eine HTML-Seite folgendermaßen schreiben:

```
<HTML>
<HEAD>
<TITLE><?php echo $titel ?></TITLE> ...
```

Eine andere Form kann aber auch so aussehen:

```
<TITLE>
<?php
  echo $titel
?>
</TITLE> ...
```

Je nach Umfang des in PHP geschriebenen Codes wird die eine oder andere Variante günstiger sein. Eine optisch eindeutige Trennung von HTML und PHP hat sich in der Praxis als sinnvoll erwiesen. Im nächsten Abschnitt wird das Einschließen von Dateien vorgestellt. Im Vorgriff auf die Befehlsbeschreibungen wird diese Form der Skripterstellung vorgezogen. Sie sollten gleich bei den ersten Projekten PHP-Funktionen in Dateien auslagern. Diese Dateien werden dann Module genannt (Modula-Programmierer wissen natürlich, dass diese Methode nicht einmal eine schlechte Imitation ist). Sie können so den Code in der HTML-Seite einigermaßen überschaubar halten. Bei größeren Applikationen vereinfacht sich die Wartung deutlich.



Hinweis

Befehle werden in PHP generell mit einem Semikolon abgeschlossen, nur wenn der Befehl allein zwischen `<?php` und `?>` steht, ist das optional.

3.1.3 Dateien einschließen

include

Mit dem Befehl `include` wird eine PHP- oder HTML-Datei eingeschlossen und so ausgeführt, als wäre sie alleine aufgerufen worden. Sie können dies auch in Schleifen einsetzen.

```
include("inc/kopf.inc.php4");
```

Dies ist eine Sprachanweisung (deshalb hier als Befehl bezeichnet, nicht als Funktion). Sie müssen einen Block bilden, um sie innerhalb anderer Anweisungen ausführen zu können. Blöcke werden in ➡ Abschnitt 3.3.1 *Blöcke und Strukturen* ab Seite 230 ausführlich betrachtet.

```
if (isset($datei)) {  
    include($datei);  
} else {  
    include("inc/notfound.inc.php4");  
}
```

Es ist möglich, dass die Befehlsstruktur in einer mit `include` eingebundenen Datei als Block aufgefasst wird. Damit ist auch die Anwendung des Befehls `return` möglich (➡ Abschnitt 3.3.4 *Benutzerdefinierte Funktionen* ab Seite 245). Der Rückgabewert wird der Funktion übergeben:

```
$value = include("modul.inc.php4");
```

require

Wenn Sie ein Modul fest in die Seite einbinden möchten, so als wäre es an dieser Stelle direkt untergebracht, nutzen Sie dagegen den Befehl `require`. Dieser ist als Sprachkonstrukt definiert.

```
require("sql_modul.inc.php4");
```

In Dateien, die mit `require` eingebunden werden, muss der PHP-Code in den Begrenzungszeichen stehen.

Die Unterschiede zwischen include und require

Der Unterschied zwischen den beiden Befehlen liegt im Zeitpunkt der Verarbeitung der einzuschließenden Datei. `require` wird zuerst ausgeführt – vor dem Parsen des gesamten Skriptes. Bei `include` dagegen erfolgt die Integration erst zu dem Zeitpunkt, wenn die Abarbeitung an dieser Stelle angelangt ist. Wird die Stelle interpretiert, kann sie natürlich auch in einer Schleife stehen. Allerdings muss `include` dann als Block gekennzeichnet sein und in geschweiften Klammern stehen.

Tipps im Umgang mit eingeschlossenen Dateien

Normalerweise können Nutzer den Inhalt der Skripte nicht sehen. Jede Datei mit der Endung PHP oder PHP4 wird vom Webserver an PHP gesendet. Es ist natürlich auch möglich, jede andere Endung anzugeben, auch HTML. Das geht bei Dateien, die mit Sicherheit keinen PHP-Code enthalten, aber zu Lasten der Leistung. Oft werden Dateien, die mit `include` eingeschlossen werden sollen, mit `.INC` bezeichnet. Auch diese Endung wird nicht verarbeitet. Das ist für den Ablauf des Skripts egal – die Verarbeitung erfolgt im Rahmen des »umgebenden« Skripts und damit unter dessen Regie.

Wenn nun aber ein Nutzer den Pfad zu den Include-Dateien herausbekommt, kann er deren Namen in der Adresszeile des Browsers direkt eingeben. Der Webserver wird die Endung nicht kennen und dem Browser die Datei direkt anbieten. Der erkennt einfachen Text und stellt ihn dar. Da in einzuschließende Dateien auch Kennwörter für Datenbanken stehen können, entsteht so eine erhebliche Sicherheitslücke.

Sicherheitsprobleme

Abhilfe schaffen Sie sehr einfach. Benennen Sie einfach alle per `include` eingeschlossenen Dateien mit dem Suffix `.INC.PHP4`. So haben Sie eine eindeutige Kennzeichnung und erzwingen im Notfall das Parsen des Codes. Das mag zwar zu einer Fehlermeldung führen – an den Inhalt gelangt der Nutzer dennoch nicht.

Abhilfe

3.1.4 PHP und JavaScript

Oft wird von der Kombination PHP-JavaScript gesprochen. Beides hat nichts direkt miteinander zu tun. JavaScript wird im Browser abgearbeitet und PHP auf dem Server. Betrachten Sie JavaScript als Erweiterung zu HTML. Der neueste Entwicklungsstand DHTML geht ohnehin von JavaScript als Anweisungssprache aus.

Selbstverständlich können Sie ebenso wie normales HTML auch JavaScript-Anweisungen dynamisch erzeugen oder mit variablen Attributen versehen. Es bleibt Ihrer Fantasie überlassen, daraus raffinierte Anwendungen zu erstellen. Anwendungsmöglichkeiten finden Sie in ➡ Abschnitt 4.1.4 *Formulare und JavaScript* ab Seite 308.

3.2 Eine kompakte Einführung in PHP

Dieser Abschnitt wendet sich vor allem an Programmieranfänger, die auch die grundlegenden Methoden der Programmierung erlernen möchten. Kenner anderer Skriptsprachen können den Abschnitt überfliegen und bei der täglichen Arbeit die Kurzreferenz zur Hand nehmen.

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

men. Für die Arbeit mit dem Buch sei außerdem das Buch *PHP 4. Die Referenz* empfohlen, ebenfalls bei Carl Hanser erschienen.

3.2.1 Ausgabe von Daten an den Browser

Wie die Ergebnisse eines Skripts zum Browser gelangen, ist sicher eine der ersten Aufgaben, mit denen Sie konfrontiert werden. Die hier gezeigten Methoden werden in allen folgenden Skripten verwendet.

Ausgabe mit echo und print

echo

Schon bei den ersten Beispielen werden Sie mit dem Befehl `echo` konfrontiert. `echo` gibt Werte an den Browser aus. Der auszugebende Text wird in Anführungszeichen geschrieben:

```
<?php echo "Dies hier wird angezeigt"; ?>
```

Innerhalb doppelter Anführungszeichen werden Variablen ausgewertet, so dass Sie sich umständliche Verkettungen sparen:

```
<?php echo "Es ist jetzt $zeit Uhr"; ?>
```

Variablen werden in ➞ Abschnitt 3.2.3 *Variablen* ab Seite 157 erläutert. Lassen Sie sich hier nicht davon irritieren, wie es funktioniert. Probieren Sie es einfach aus. Wenn Sie diese Möglichkeit unterdrücken möchten, schreiben Sie einfache Anführungszeichen:

```
<?php echo 'Das Buch kostet $59.'; ?>
```

print()

Alternativ kann die Funktion `print` zur Ausgabe verwendet werden.

```
<?php print ("Es ist jetzt $zeit Uhr"); ?>
```

Die Unterschiede zwischen echo und print

Im Gegensatz zu `echo` ist `print` eine Funktion. Es gibt Fälle, in denen die Ausgabe mit einer Rückgabe gekoppelt werden muss, beispielsweise beim ternären Bedingungsoperator:

```
$a == 0 ? print "a ist 0" : print "a ist nicht 0";
```

Diese Konstruktion wertet eine Bedingung aus und führt entsprechend dem Ergebnis den ersten oder zweiten Teil aus. Es kommt zwar nur auf die Ausgabe an, die Konstruktion erwartet aber Rückgabewerte – der Raum zwischen `?` und `:` darf aus syntaktischen Gründen nicht »leer« sein. Hier kann also niemals `echo` stehen. Allerdings geben Ausdrücke immer etwas zurück, weswegen auch diese Schreibweise erlaubt wäre:

```
echo $a == 0 ? "a ist 0" : "a ist nicht 0";
```

`print` darf nur ein Argument haben. Wenn Sie mehrere Werte haben, nutzen Sie die Schreibweise mit Anführungszeichen oder setzen eine

Zeichenkette mit dem Operator `.` (Punkt) zusammen. Hier ist `echo` flexibler, es sind beliebig viele Argumente erlaubt, die jeweils durch ein Komma getrennt werden:

```
<?php echo "Heute ist", $wochentag, "!"; ?>
```

Der einzige Unterschied zu der Verknüpfung der Zeichenketten dürfte die Geschwindigkeit sein. In der Praxis ist die Ausgabe aber ohnehin der schnellste Teil des Skripts. Damit bleibt als einziges Argument der optische Eindruck:

```
<?php echo "Heute ist" . $wochentag . "!"; ?>
```

Ausgabe langer Texte mit der heredoc-Syntax

Aus Perl ist der so genannte »heredoc«-Operator⁶ bekannt. In Perl besteht das Problem, dass dies keine eingebettete Skriptsprache, sondern eine reine Skriptsprache ist. Damit müssen alle Ausgaben mit `echo` erzeugt werden, eine unter Umständen zeitraubende Prozedur. Die Anwendung ist einfach. Am Beginn des Blocks wird eine Zeichenkette definiert, die am Ende wiederholt wird. Länge und Inhalt sind völlig frei, müssen aber so gewählt werden, dass der Text sich nicht im Block wiederholt.

Hier ein Beispiel:

```
echo <<<TABLEHEAD
<table>
  <tr>
    <th>Name</th>
    <th>Adresse</th>
  </tr>
</table>
TABLEHEAD;
```



Entscheidend ist der Operator `<<<`, der die Sequenz einleitet. Bei Perl ist es übrigens der Operator `<<`, der aber in PHP eine andere Bedeutung hat. Zwischen Operator und Name des Blocks steht kein Leerzeichen. Der Text zwischen den Markierungen wird unverändert ausgegeben, also inklusive aller Leerzeichen, Tabulatoren oder Zeilenumbrüche, die – egal aus welchem Grund – mit eingetippt wurden. Das spielt bei HTML aber nur selten eine Rolle.

Die Anwendung ist nicht auf den `echo`-Befehl beschränkt. Sie können auch einer Variablen einen solchen Block übergeben:

```
$addresses = <<<ADRESSEN
Martin Meier,
```



⁶ Allerdings ist die Schreibweise in Perl etwas anders, nur das Prinzip ist übernommen wurden.

```
Lieschen Müller,  
Clemens Krause,  
Ottokar Domma  
ADRESSEN;
```

Das sieht unter Umständen viel übersichtlicher und einfacher aus, als wenn die Werte hintereinander stehen. Denken Sie aber daran, dass hier die Zeilenumbrüche erhalten bleiben – nicht in jedem Fall funktioniert das problemlos.

Typische Probleme Am Anfang passiert es oft, dass der Parser nach dem Einbau der Blöcke Fehler meldet. Das folgende Beispiel funktioniert nicht wie erwartet:

```
if ($strFeld == 1) {  
    $addresses = <<<ADRESSEN  
    Martin Meier,  
    Lieschen Müller,  
    Clemens Krause,  
    Ottokar Domma  
    ADRESSEN;  
}
```

Der Fehler liegt nicht direkt im Code, sondern in der Art, wie PHP diesen verarbeitet. Das Ende der Sequenz wird nur erkannt, wenn das Schlüsselwort *ADRESSEN* am Anfang der Zeile steht. Durch die Einrückung – egal ob mit Tabulator oder Leerzeichen – wird es nicht erkannt.



In einigen Versionen konnte beobachtet werden, dass auch Leerzeichen hinter dem letzten Semikolon die Erkennung des Endes des Blocks unterdrücken – eindeutig ein Bug in PHP. Falls Sie Syntaxfehler erhalten und sich sicher sind, dass die Schreibweise der hier gezeigten Darstellung entspricht, entfernen Sie diese Leerzeichen bzw. Tabulatoren.

3.2.2 Kommentare

**Warum
Kommentare
wichtig sind**

Jedes Programm, auch das kleinste, sollte sauber kommentiert werden. Eine ordentliche Dokumentation erleichtert die Arbeit während der Entwicklung erheblich. Kommentare sind ein wesentlicher Bestandteil der Dokumentation. Stil und Inhalt sollten dem Programmierer die Übersicht in seinem Code erhalten, aber auch anderen mit dem Programm beschäftigten Personen die Lesbarkeit ermöglichen. Denken Sie daran, dass Ihre Überlegungen bei der Abstraktion eines Problems nicht immer ohne weiteres nachvollziehbar sind. Oft gibt es sehr viele Lösungen für eine Aufgabe. Warum Sie eine bestimmte Lösung gewählt haben und wie diese anzuwenden ist, wird in Kommentaren beschrieben.

Bei der Gestaltung von HTML-Seiten sind Sie mit Kommentaren sicher schon in Berührung gekommen. HTML-Kommentare beginnen mit den Zeichen `<!--` und enden mit `-->`. Sie können sich über mehrere Zeilen erstrecken:

```
<!--  
Hier wird der HTML-Quelltext kommentiert  
-->
```

Da der PHP-Code direkt in die HTML-Seiten eingebunden wird, könnte man auch diese Methode zur Kommentierung verwenden. In der Praxis sollten Sie dies nicht tun, denn damit wären die Kommentare für jeden Nutzer der Website sichtbar. Auch wenn Ihnen das egal ist – Sie würden viel wertvolle Bandbreite im Web verschwenden, wenn die Kommentare zum Browser übertragen werden.

Kommentare im PHP-Code werden nicht mit übertragen. Der Parser ignoriert diese Zeilen und entfernt Sie vor der Übertragung der Seite zum Browser. Kommentare in PHP beginnen mit `/*` und enden mit `*/`. Sie können diese Schreibweise für alleinstehende Kommentare, Kommentare über mehrere Zeilen oder auch für Kommentare am Zeilenende verwenden:

```
<html><body>  
<h1>Kommentare</h1>  
<?php  
/* Erste Ausgabe */  
echo ("Mein erstes PHP-Skript<br>");  
/*  
Zweite  
Ausgabe  
*/  
echo ("Mein zweites PHP-Skript<br>"); /* 3. folgt */  
echo ("Mein drittes PHP-Skript<br>");  
?>  
</body></html>
```

HTML-Kommentare

/ ... */*

Listing 3.1:
Beispiel comment_1:
Kommentare

Gegenüber VBScript, wo Kommentare auf der Zeile nur eingeleitet werden müssen, mag diese Schreibweise aufwändiger erscheinen. Sie können dies aber zu Ihrem Vorteil ausnutzen, indem Sie Kommentare mitten in der Zeile platzieren.

```
<html>  
<body>  
<h1>Kommentare (2)</h1>  
<?  
echo /* wirklich? */ ("Mein Tausendstes PHP-Skript");  
?>  
</body>  
</html>
```

Listing 3.2:
Beispiel comment_2:
Kommentare in der
Zeile

Fehlerquellen

Die Variante aus Listing 3.2 sollten Sie nicht zu umfangreich nutzen, denn darunter leidet die Übersichtlichkeit. Außerdem können sich leicht typische »Anfängerfehler« einschleichen, wie es der folgende Code zeigt, der tatsächlich keinen Kommentar enthält:

```
<?php
echo ("Mein Tausendstes /* wirklich? */ PHP-Skript");
?>
```

//

Der PHP-Parser kennt auch die aus Java und C bekannten Kommentarteichen //. Sie sollten diese nicht als ganzzeilige Kommentare verwenden. HTML-Seiten werden oft mit DHTML oder JavaScript ausgestattet. In beiden Fällen werden JavaScript-Funktionen definiert. Diese werden vor alten Browsern oft versteckt und verwenden dazu die beiden Schrägstriche als Kommentar:

*Listing 3.3:
Beispiel comment_3:
Drei Kommentar-
arten in einem
kleinen Skript*

```
<SCRIPT LANGUAGE="JavaScript">
<-- Beginn verstecken
function tuwas()
{
    /* Hier wird etwas in JavaScript erzeugt */
    alert('Ich tu was');
    /* Ende der Funktion */
}
// Ende verstecken -->
</SCRIPT>
```

Es ist also gut möglich, dass Sie mit drei verschiedenen Sprachen in einer Datei arbeiten und damit auch über drei Arten von Kommentaren verfügen. Es ist eine gute Angewohnheit, hier nichts durcheinander zu bringen, auch wenn die Parser dies nicht erzwingen.

Dennoch bietet sich eine gute Einsatzmöglichkeit des //-Kommentars an. Wenn Sie auf jeder Zeile eine kurze Bemerkung unterbringen möchten, schreiben Sie diese am Zeilenende auf:

```
$zahl = wert($value, "string") // formatiert Variable $value
```

Ein expliziter Abschluss dieser Kommentarart ist nicht notwendig.

#

Für den letzten Fall gibt es noch eine weitere Variante. Mit dem Zeichen # können ebenfalls Kommentare gekennzeichnet werden:

```
$zahl = wert($value, "string") # Beginn Fall 3
```

Was Sie letztlich verwenden, bleibt Ihrem Geschmack überlassen. Entscheidend ist die Lesbarkeit, vor allem für andere Leser des Codes. Übertreiben Sie die Kommentierung jedoch nicht. Eindeutige Zeilen sollten nicht mit Zusatzinformationen überfrachtet werden.

3.2.3 Variablen

In jedem Skript kommen Sie schnell in die Verlegenheit, Werte (Daten) speichern zu müssen. Für die Speicherung während der Laufzeit des Skripts stehen Ihnen Variablen zur Verfügung. Intern sind Variablen Namen, die auf einen bestimmten Speicherplatz im Computer verweisen, an dem der Inhalt der Variable gespeichert ist. Wie der Name Variable andeutet, kann der Inhalt mit vielfältigen Maßnahmen gewandelt werden – er ist variabel.

Einer Variablen einen Wert zuweisen

Um eine Variable zu erzeugen, wird ihr einfach mit dem Operator = ein Wert zugewiesen. Dass es sich um eine Variable handelt, erkennt PHP anhand des vorangestellten \$-Zeichens (analog der Perl-Syntax):

Variablen werden durch ein \$ eingeleitet

```
$name = "Jörg Krause";  
$i = 1;  
$zahl = 14.95; // Gleitkommazahlen werden mit Punkt geschrieben
```

Wenn Sie andere Programmiersprachen kennen, fällt dabei auf, dass kein Typ für die Variablen vorgegeben wurde. Sie finden Möglichkeiten der Typvergabe und der Typumwandlung in ➡ Abschnitt 3.2.4 *Datentypen* ab Seite 161.

PHP-Variablen können typlos sein

Der Typ wird bei der ersten Zuweisung der Variablen erkannt. Spätere Zuweisungen von Werten werden dann mit Hilfe einer automatischen Typumwandlung vorgenommen. Im Beispiel wird der Variablen *\$name* der Typ string (Zeichenkette) zugeordnet, *\$i* wird als integer (Ganzzahl) und *\$zahl* als double (Fließkommawert) erkannt.

Gültigkeitsbereiche von Variablen

Variablen sind in PHP nur in dem Kontext gültig, in dem sie definiert wurden. Sie werden in ➡ Abschnitt 3.3.4 *Benutzerdefinierte Funktionen* ab Seite 245 näher kennen lernen, was sich dahinter verbirgt. Wenn Sie Variablen innerhalb eines solchen Funktionsblocks anlegen, werden sie außerhalb nicht mehr sichtbar sein. Umgekehrt gilt dies auch; global (außerhalb der Funktion) angelegte Variablen sind nicht ohne weiteres innerhalb einer Funktionsdefinition sichtbar. Das folgende Beispiel zeigt, wie es nicht funktioniert:

```
<?php  
$zahl = 22;  
function ausgabe()  
{  
    echo $zahl;  
}  
ausgabe()  
?>
```

Listing 3.4:
Dieses Beispiel funktioniert nicht wie erwartet

An dieser Stelle sei angemerkt, dass es sich um einen Block handelt, den Sie an den umgebenden geschweiften Klammern { } erkennen. Die Variable `$zahl` ist hier nicht bekannt. Blöcke treten allerdings auch im globalen Teil eines Skripts auf und sind alleine kein Bereich, der die Gültigkeit von Variablen einschränkt. Dies gilt tatsächlich nur für Funktions- und Klassendefinitionen.

global **\$GLOBALS["x"]**

Der explizite Zugriff auf globale Variablen ist dennoch mit dem Schlüsselwort `global` möglich. Das in Listing 3.4 gezeigte Beispiel funktioniert, wenn Sie PHP mitteilen, dass es sich um eine globale Variable handelt.

Listing 3.5:
*Beispiel `var global`:
Globale Variablen
anwenden*

```
<?php
$zahl = 22;
function ausgabe() {
    global $zahl;
    echo "Und hier kommt die Zahl: $zahl";
}
ausgabe();
?>
```

Mit dem Schlüsselwort `global` teilen Sie PHP mit, dass die Variable in einem globalen Kontext bereits existiert. Der Inhalt der Variable wird dann übernommen. Umgekehrt werden sich Änderungen an der Variablen innerhalb des Blocks auch `global` widerspiegeln. Sie können hinter `global` mehrere Variablen schreiben, durch Kommata getrennt:
`global $x, $y, $zahl;`

Intern verwaltet PHP die Variablen in einem Array (mehr darüber erfahren Sie in ➡ Abschnitt 3.2.6 *Operatoren* ab Seite 170). Das letzte Beispiel könnte man damit auch so schreiben:

Listing 3.6:
*`var global2`: Andere
Variante des Zugriffs
auf globale Variablen*

```
<?php
$zahl = 22;
function ausgabe() {
    echo "Und hier kommt die Zahl: ", $GLOBALS["zahl"];
}
ausgabe();
?>
```

Beachten Sie hier, dass der Parameter des Arrays der Name der Variablen *ohne* das führende `$`-Zeichen ist. Der Zugriff auf das gesamte Array `$GLOBALS[]` ist indes nicht möglich.

static

Variablen innerhalb eines Gültigkeitsbereiches sind flüchtig – beim Verlassen des Blocks werden sie gelöscht, beim erneuten Eintritt wieder angelegt. In solchen Fällen müssten Sie immer auf globale Variable zurückgreifen, was bei großen Projekten zu einem unüberschaubaren Chaos an Variablennamen führen würde. Der professionellere Weg sind Parameterübergaben, die in ➡ Abschnitt 3.3.4 *Benutzerdefinierte Funktionen* ab Seite 245 beschrieben werden. Mit dem Schlüsselwort

static können Sie Variablen innerhalb eines Blocks so definieren, dass sie erhalten bleiben, wie das folgende Beispiel zeigt:

```
<?php
function ausgabe()
{
    static $zahl = 22;
    echo "Und hier kommt die Zahl: $zahl<br>";
    $zahl++;
}
ausgabe();
ausgabe();
ausgabe();
?>
```

Listing 3.7:
Beispiel var static:
Statische Variablen

Der Operator ++ erhöht die Variable jedes Mal um eins. Die Zuweisung des Startwerts erfolgt nur beim ersten Aufruf der Funktion, die drei Aufrufe führen also tatsächlich zur Ausgabe von 22, 23 und 24.

Eine interessante Anwendung sind rekursive Funktionen, bei denen sich die Funktion unter bestimmten Umständen selbst aufruft. Ein praktisches Beispiel finden Sie in ➡ Abschnitt 3.3.4 *Benutzerdefinierte Funktionen* ab Seite 245.

Dynamische Variablen

Interessant ist die Möglichkeit in PHP, auch Variablennamen selbst in Variablen zu speichern und so quasi implizit auf Variablen zuzugreifen.

In einige Veröffentlichungen werden diese Variablen auch als »variable Variablen« bezeichnet, was zwar eher der direkten Übersetzung aus dem englischen Handbuch entspricht, aber nicht ganz das Verhalten widerspiegelt und zudem zungenbrecherisch wirkt. In diesem Buch wird deshalb konsequent von dynamischen Variablen gesprochen.



Die Zuweisung erfolgt in zwei Schritten. Zuerst wird eine normale Variable erzeugt, der Sie den Namen der dynamischen Variablen zuordnen:

```
$vari = "name";
```

Eine dynamische Variable nimmt den Wert einer Variablen als Namen. Der impliziten Variablen können Sie einen Wert zuweisen, indem zwei \$\$-Zeichen vorangestellt werden:

\$\$variable

```
$$vari = "PHP";
```

Wenn Sie die zuvor gezeigte Definition verwendet haben, gibt das Skript nun mit echo(\$name); den Wert »PHP« aus. Einmal erfolgte Zuweisungen bleiben von späteren Umbenennungen der führenden Variablen unberührt. Das folgende Listing zeigt einige Varianten:

Listing 3.8:
Beispiel var variabel:
Dynamische
Variablen

```
<?php
$vari = "name";
echo ("Variable 'vari' ist: $vari <BR>");
$name = "PHP";
echo ("Variable 'name' ist: $name <BR>");
$$vari = "Dynamic";
echo ("Variable 'name' ist: $name <BR>");
$vari = "sprache";
echo ("Variable 'vari' ist: $vari <BR>");
echo ("Variable 'sprache' ist: $sprache <BR>");
?>
```

Führen Sie das Beispiel aus und schauen Sie sich die Ausgabe genau an. Nehmen Sie sich die Zeit, ein paar Minuten darüber nachzudenken und mit dem Code zu spielen, um die Wirkungsweise der dynamischen Variablen kennen zu lernen.

Dynamische Variablen mit Arrays

Dynamische Variablen können auch mit Arrays genutzt werden. In diesem Fall kann es zu Zuordnungsproblemen kommen. Der PHP-Parser kann nicht immer eindeutig erkennen, auf welchen Teil der Konstruktion sich die Indizes beziehen. So könnte mit \$\$zahl[1] sowohl zahl[1] eine Variable sein oder \$zahl als Teil eines Arrays mit dem Index 1. In solchen Fällen fassen Sie die logisch zusammenhängenden Teile des Ausdrucks mit geschweiften Klammern zusammen, also entweder \${zahl[1]} oder \${zahl}[1].

Anwendungsmöglichkeiten für dynamische Variablen

Diese komplizierte Art der Verarbeitung von Variablen mag am Anfang wenig sinnvoll erscheinen. Das folgende Beispiel zeigt, wie es angewendet werden kann.

Angenommen, Sie möchten sich Variablen über mehrere Skripte hinweg merken. Das kann in einer Datei oder einer Datenbank geschehen, darauf wird später noch eingegangen. In jedem Fall speichern Sie die zu speichernden Variablen in einem Array (siehe ➡ Abschnitt 3.2.6 Operatoren ab Seite 170), wobei immer der Name der Variablen und der Inhalt zusammen ein Element ergeben.

Listing 3.9:
var gen: Auto-
matisierte Erzeugung
von Variablen

```
<?php
$arrUS = array("name" => "Krause",
               "alter" => 37,
               "ort" => "Berlin");
if (is_array($arrUS)) {
    foreach($arrUS as $var => $val) {
        ${$var} = $val;
    }
}
echo "$name wohnt in $ort und ist $alter Jahre alt";
?>
```

Wenn dieses Skript abläuft, entstehen drei Variablen: `$name`, `$alter` und `$ort`, gefüllt mit den Daten "Krause", 37 und "Berlin". Stellen Sie sich vor, anstatt des Arrays eine serialisierte Zeichenkette zu haben – dann steht der Speicherung im Tabellenfeld einer Datenbank nichts im Wege.

Der Trick besteht in der Verwendung des Variablennamenoperators. Die entscheidende Zeile zeigt, wie aus der Zeichenkette eine gleichlautende Variable wird:

```
${$var} = $val;
```

Referenzen

Mit Hilfe von Referenzen kann ein Verweis auf eine Variable erstellt werden. Dabei wird der Wert nicht kopiert, sondern bleibt nur an der ursprünglichen Stelle bestehen. Ändert sich später diese Quelle, wirkt sich das auf alle Referenzen aus. Der folgende Code zeigt dies:

```
<?php
$zahl = 14;
$ziel = &$zahl;
$zahl = 15;
echo $ziel;
?>
```

Listing 3.10:
var ref: Referen-
zierung von
Variablen

Das Skript gibt die Zahl 15 aus. Die Änderung an `$zahl` wirkt sich direkt auf die Referenz `$ziel` aus.

Referenzen entstehen, indem der Quellvariablen das Zeichen `&` vorangestellt wird. **&\$**

3.2.4 Datentypen

Im letzten Abschnitt wurde bereits angedeutet, dass PHP weitestgehend typlos arbeitet. Sie müssen also Variablen vor der ersten Verwendung nicht mitteilen, um welchen Typ es sich handelt. PHP wird, wann immer es möglich ist, Datentypen selbst erkennen und umwandeln. Sie können dieses Verhalten aber vielfältig beeinflussen.

Die Datentypen

PHP kennt die in Tabelle 3.1 gezeigten Datentypen für Variablen.

Schlüsselwort	Beschreibung
integer, int	Ganzzahl, int ist eine alternative Schreibweise für integer.
boolean, bool	Logischer Wert (TRUE oder FALSE), intern 0 oder 1, bool ist eine alternative Schreibweise

Tabelle 3.1:
Datentypen

Schlüsselwort	Beschreibung
double	Fließkommazahl
real	Eine alternative Schreibweise für double
string	Zeichenkette mit 0 bis 32 768 Zeichen
array	Array
object	Objekt

Die Möglichkeiten sind begrenzt, verglichen mit Programmiersprachen, für die Praxis aber ausreichend. Trotzdem sollten Sie sich über den (internen) Typ einer Variablen immer im Klaren sein. Es erleichtert die Fehlersuche erheblich, wenn Sie Zuordnungen von Typen nicht dem Zufall überlassen.

resource

Intern wird noch der Typ resource erkannt, der aber nicht als Basis für andere Datentypen in Frage kommt. Damit werden Zeiger (sogenannte Handle) auf Datenquellen verwaltet.

Automatische Typumwandlung

Regeln der automatischen Typumwandlung

Bei der automatischen Typumwandlung versucht PHP zuerst, die rechte Seite des Ausdrucks zu erkennen und dann der linken Seite zuzuweisen. Dabei geht PHP nach bestimmten Regeln vor:

- Werden Werte in ""-Zeichen eingeschlossen, wird eine Zeichenkette angenommen.
- Wird eine Zahl ohne Punkt (Komma) geschrieben, wird eine Ganzzahl angenommen.
- Wird eine Zahl mit Punkt (Komma) gefunden, wird eine Fließkommazahl angenommen.

Listing 3.11:
var_datatypes:
Automatische
Typumwandlung

```
<?php
$variable = "17";
echo ++$variable;
echo '<br>';
$variable = 17;
echo ++$variable;
echo '<br>';
$variable = 17;
$variable = $variable + 23.5;
echo $variable;
echo '<br>';
$variable = 12;
$variable = $variable + "5 Stück";
echo $variable;
echo '<br>';
$variable = "12";
$variable = $variable . "5 Stück";
```

```
echo $variable;
?>
```

Bei der Auswertung von Ausdrücken bestimmt der verwendete Operator den Typ des Gesamtausdrucks. Wenn dadurch ein numerischer Typ erzwungen wird, versucht PHP einen folgenden Zeichenkettenausdruck nach Zahlen zu durchsuchen. Im vorletzten Beispiel wird die Zeichenkette "5 Stück" so zu 5 gewandelt. Im letzten Beispiel wird dagegen 125 Stück als Ergebnis ausgegeben. Der Punkt ist der Operator für die Verknüpfung von Zeichenketten.

Typkonvertierung

In vielen Fällen werden Sie mit Datenbanken arbeiten, die den Umgang mit Datentypen nicht so locker sehen. Bevor Sie sich mit den vielfältigen Fehlermeldungen auseinander setzen, wandeln Sie die Typen gleich explizit um. Sie können die bereits in

(integer), (int)
(string)
(double), (real)
(array)
(object)
(boolean), (bool)

Tabelle 3.1 erwähnten Datentypen einem Ausdruck voranstellen und damit den Zieltyp erzwingen:

```
<?php
$variable = 100;
$fvar = (double) $variable + "100.5";
$svar = (int) $fvar;
echo ("Double: $fvar , Integer: $svar");
?>
```

Listing 3.12:
*var*types forced:
Explizite Typumwandlung

Da die Typumwandlung mit Operatoren Priorität hat, wird der Einsatz nur da zwingend sein, wo es keine eindeutigen Operatoren gibt, wie in Listing 3.12 bei der Umwandlung einer Fließkommazahl in eine Ganzzahl.

Bei der Umwandlung sollten Sie bedenken, dass nicht alle Richtungen sinnvoll sind. Tabelle 3.2 zeigt die möglichen Kombinationen.

Zieltyp	Sinnvolle Quelltypen	Umwandlungsprinzip
integer int	double	Dezimale werden abgeschnitten (keine Rundung)
	string	Wird keine Zahl erkannt, wird 0 zurückgegeben
	boolean	TRUE wird zu 1 und FALSE zu 0
double real	integer	unverändert
	string	Wird keine Zahl erkannt, wird 0 zurückgegeben

Tabelle 3.2:
Explizite Umwandlung der Datentypen

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Zieltyp	Sinnvolle Quelltypen	Umwandlungsprinzip
string	integer double	Gibt die Zahl als Zeichenkette zurück
array	object	Wird direkt umgewandelt
	integer string double	Es entsteht ein Array mit einem Element vom Ursprungstyp
object	array	Wird direkt umgewandelt
	integer string double	Es entsteht ein Objekt mit einer Eigenschaft, die durch die Variable repräsentiert wird.
boolean	integer	0 wird FALSE, jede andere Zahl wird TRUE
	double	0 wird FALSE, jede andere Zahl wird TRUE

Das Verhalten bei der Umwandlung in array und object wird in den entsprechenden Abschnitten noch näher gezeigt.

Zeichenketten- konvertierung

Bei der Umwandlung von Zeichenketten wendet PHP bestimmte Regeln an. Mit deren Kenntnis können Sie das Ergebnis voraussagen und von der Umwandlung sicher Gebrauch machen:

- double entsteht, wenn die Zeichenkette mit einem gültigen numerischen Zeichen (Ziffer, Plus oder Minus) beginnt und dahinter die Zeichen `».«`, `»e«` oder `»E«` folgen. Ist der erste Teil der Zeichenkette ein gültiger Ausdruck, wird der Rest ignoriert.
- integer entsteht, wenn die Zeichenkette mit einem gültigen numerischen Zeichen (Ziffer, Plus oder Minus) beginnt und dahinter nicht die Zeichen `».«`, `»e«` oder `»E«` folgen. Ist der erste Teil der Zeichenkette ein gültiger Ausdruck, wird der Rest ignoriert.



Hinweis

Die Zeichen `e` oder `E` dienen der Darstellung von Exponenten. Die Schreibweise `Enn` steht für $\times 10^{nn}$. Generell dient der Punkt `.` als Dezimaltrennzeichen (amerikanische Schreibweise). Sie müssen dies bei Zuweisung von Variablen berücksichtigen. Zur Ausgabe lassen sich Zahlen mit der Funktion `number_format` in die benötigte (auch deutsche) Form bringen.

Umwandlungsfunktion

settype()

Neben der Angabe des Datentyps in runden Klammern kann auch die Funktion `settype` genutzt werden. In manchen Ausdrücken wird eine Funktion erwartet, oft dient der Einsatz nur der besseren Lesbarkeit.

Wenn Sie keinen Ausdruck benötigen, spart der Einsatz sogar Schreibarbeit:

```
settype($variable, "integer");
```

Die Funktion gibt einen numerischen Wert zurück. Dieser Wert ist 1 (TRUE), wenn die Umwandlung erfolgreich war, sonst 0 (FALSE). Als erster Parameter wird die Variable angegeben, die umzuwandeln ist. Der zweite Parameter ist der Typ in Form einer Zeichenkette.

Wenn Ihnen `settype` zu umständlich ist, können Sie auch die abgeleiteten *val*-Funktionen verwenden:

- `intval($variable)`

Diese Funktion wandelt in integer um.

- `doubleval($variable)`

Diese Funktion wandelt in double um.

- `strval($variable)`

Diese Funktion wandelt in string um.

intval()
doubleval()
strval()

Datentypen bestimmen

Da PHP Umwandlungen selbst vornimmt, ist es manchmal wichtig zu wissen, welcher Typ intern genutzt wird. Sie können dazu die Funktion `gettype` einsetzen:

```
echo "Der Typ ist: ", gettype($variable);
```

Die Funktion gibt die bereits bekannten Typbezeichner als Zeichenkette zurück. Konnte der Typ nicht erkannt werden, so wird die Zeichenkette "unknown type" erzeugt.

Bei logischen Ausdrücken ist die Verwendung von `gettype` zu umständlich. Sie können deshalb eine ganze Reihe von Funktionen einsetzen, die 1 (TRUE) oder 0 (FALSE) zurückgeben:

gettype()

is_integer()
is_double()
is_string()
is_array()
is_object()
is_bool()

Funktion	TRUE bei folgendem Datentyp
<code>is_long(\$variable)</code> <code>is_integer(\$variable)</code>	integer
<code>is_double(\$variable)</code> <code>is_real(\$variable)</code>	double
<code>is_string(\$variable)</code>	string
<code>is_array(\$variable)</code>	array
<code>is_object(\$variable)</code>	object
<code>is_bool(\$variable)</code>	boolean

*Tabelle 3.3:
Testfunktionen für
Datentypen*

is_scalar() Eine etwas allgemeinere Funktion ist `is_scalar`, mit der Basisdatentypen erkannt werden (in der Fachsprache werden diese Skalare genannt). Dies sind die Datentypen `integer`, `float`, `string` und `boolean`. Arrays und Objekte sind dagegen zusammengesetzte Datentypen – also keine Skalare.

Spezielle Variablenzustände bestimmen

is_resource()
get_resource_type() Auch für die spezielle Form `resource` gibt es eine Testfunktion: `is_resource`. Damit kann beispielsweise nach einem Datei- oder Datenbankzugriff ermittelt werden, ob dieser erfolgreich verlief. Derartige Funktionen geben Handler zurück, die PHP in späteren Funktionsaufrufen verwendet, um wieder Verbindung mit der Quelle aufzunehmen. Wenn Sie nicht nur wissen wollen, ob eine Ressource existiert, sondern auch den Typ benötigen, hilft `get_resource_type` weiter. Diese Funktion gibt eindeutige Zeichenketten wie »file« oder »mysql link« zurück.

is_null() Manchmal haben Variablen den Wert `NULL`. Sie sind dann zwar schon angelegt, aber völlig leer. Leer ist ein anderer Zustand als eine leere Zeichenkette (die ja immerhin schon eine Zeichenkette ist) oder die Zahl 0 (die eindeutig eine Zahl ist, also mehr als nichts). Der Unterschied wird bei einfachen Vergleichen mit `==` nicht immer klar, weil hier interne Typumwandlungen stattfinden und nur Werte, nicht jedoch Typen betrachtet werden. So ist der folgende Ausdruck wahr:

```
$int = 0;
if ($int == NULL) echo "wahr";
```

Das ist aber eigentlich nicht korrekt, denn `NULL` ist ein anderer Zustand als 0, wie eben beschrieben. Der Operator `===` bezieht deshalb den Typ mit ein und entscheidet korrekt, dass 0 nicht gleich `NULL` ist. Mehr zu Operatoren finden Sie in ➡ Abschnitt 3.2.6 *Operatoren* ab Seite 170.

Variablen auf Existenz testen

isset() Bei logischen Vergleichen kann es entscheidend sein, ob eine Variable mit 0 oder einer leeren Zeichenkette gefüllt ist, oder überhaupt noch nicht zugewiesen wurde. Sie können mit der Funktion `isset` testen, ob eine Variable existiert:

```
$exists = isset($variable);
```

empty() Wenn die Variable schon existiert, wird 1 (`TRUE`) zurückgegeben, sonst 0 (`FALSE`). Wenn Sie nur testen möchten, ob ein Wert zugewiesen wurde, ist die Funktion `empty` zu verwenden:

```
$leer = empty($variable);
```

Diese Funktion gibt 1 (TRUE) zurück, wenn der Inhalt der Variablen 0 oder eine leere Zeichenkette ist und die Variable existiert.

In diesem Zusammenhang besteht auch die Möglichkeit, eine Zuweisung wieder aufzuheben und eine Variable so verschwinden zu lassen:

unset()

```
unset($variable);
```

Zustand nach	if (function(\$str)) { ... }				if (formel) { ... }			
	empty	!empty	isset	!isset	\$str=="	\$str!="	\$str	!\$str
unset(\$str)	true	false	false	true	true	false	false	true
\$str=""	true	false	true	false	true	false	false	true
\$str=0	true	false	true	false	true	false	false	true
\$str='0'	false	true	true	false	false	true	false	true
\$str='x'	false	true	true	false	false	true	true	false
\$str=1	false	true	true	false	false	true	true	false
\$str=true	false	true	true	false	false	true	true	false
\$str=false	true	false	true	false	true	false	false	true
\$str=array()	true	false	true	false	false	false	false	true
\$str=array('x')	false	true	true	false	false	false	true	false
class foo; \$str=new foo;	true	false	true	false	false	false	false	true
class foo { var \$a }; \$str=new foo;	false	true	true	false	false	false	true	false

Tabelle 3.4:
Verhalten der
Funktionen empty
und isset

Der Umgang mit den Funktionen empty und isset mag auf den ersten Blick trivial erscheinen, kann aber zu kniffligen Programmierfallen führen. Der Grund liegt in der simplen internen Darstellung der Variableninhalte in PHP.

So funktioniert es richtig!

In Tabelle 3.4 finden Sie alle erdenklichen Kombinationen von Zuständen für Variablen und der Reaktion der Funktionen empty und isset. Dies sieht vielleicht verwirrend aus, wird aber klar, wenn man sich vor Augen hält, was PHP intern darstellt. So ist der Zustand TRUE, der einer Variablen zugewiesen werden kann, identisch mit folgender Zuweisung:

```
$str = 1;
```

Um sicher zu gehen, ob die Variable als Inhalt FALSE zurückgibt oder tatsächlich nicht definiert ist, eignet sich dieser Ausdruck:


```
if !( isset($str) ) { ... }
```

Zu empty ist zu bemerken, dass numerische Variablen als »leer« angesehen werden, wenn der Inhalt 0 ist. Als Zeichenkette ist aber '0' eindeutig nicht leer. Finden hier Konvertierungen statt, deren Ursache im Programmkontext zu suchen ist, kommt es zu schwer zu identifizierenden Ablauffehlern (weil die Ursache in den wechselnden Daten liegt). Die Tabelle zeigt, wie Sie »Nullsicher« testen.

3.2.5 Konstanten

Konstanten sind im Grunde nicht mehr als Speicherplätze, deren Inhalt nach der ersten Zuweisung nicht mehr verändert werden kann.

Wozu Konstanten benötigt werden

Konstanten sind in allen Programmier- und Skriptsprachen nützlich, um feste, immer wieder benötigte Werte mit verständlichen Begriffen zu umschreiben. Der Einsatz erhöht die Lesbarkeit des Programms erheblich. Außerdem werden Änderungen leichter durchführbar. Stellen Sie sich vor, Sie haben an 100 Stellen im Programm eine Verbindungsinformation zu einer Datenbank stehen. Jetzt ändern sich die Verbindungsparameter geringfügig. Sie müssten nun an 100 Stellen den Code ändern, ein langwieriges und fehleranfälliges Unterfangen. Sie können nicht einfach »Suchen und Ersetzen« verwenden, da die Zeichenfolgen auch in anderem Zusammenhang auftreten könnten.

Konstanten deklarieren

define()

Konstanten werden mit dem Schlüsselwort `define` deklariert. Der Typ ergibt sich aus den für Variablen geltenden Ausführungen:

```
define("DSN", "DSN=shop");
```

Sie können auf die so erzeugte Konstante wie auf Variablen zugreifen:

```
echo ("Die aktuelle DSN ist: ", DSN);
```

Es ist empfehlenswert, Konstanten mit Großbuchstaben zu bezeichnen. PHP selbst hat keine Konventionen für die Benennung. Es ist im Quelltext aber ungemein hilfreich, Konstanten so zu erkennen. Das fehlende `$`-Zeichen allein ist weniger auffällig.

Konstanten können nachträglich (zur Laufzeit) nicht verändert werden, der folgende Ausdruck ist unzulässig:

```
DSN = "DSN=shop";
```

defined()

Manchmal ist es notwendig festzustellen, ob eine bestimmte Konstante definiert wurde. Dazu können Sie die Funktion `defined` einsetzen, die

TRUE zurückgibt, wenn unter dem angegebenen Namen eine Konstante definiert wurde:

```
if ( defined("DSN") ) {  
    # Aktion  
}
```

Bei der Definition von Konstanten passiert es oft, dass diese folgendermaßen geschrieben wird:

```
define(FARBE, '#eefdd');
```

Dies funktioniert meist, aber eben nicht immer. Tatsächlich läuft hier ein komplexer Vorgang ab. PHP erkennt die Zeichenfolge »FARBE« und versucht, diese als Konstante zu verarbeiten. Diese Konstante existiert aber zu diesem Zeitpunkt noch nicht. Also wird eine interne Typkonvertierung vorgenommen und die Zeichenkette FARBE gebildet. Diese wird der Funktion define übergeben und die Konstante wird erstellt. Wenn statt FARBE aber die Zeichenfolge ECHO verwendet wird, reagiert PHP mit einem Fehler. Denn hier wird die Zeichenfolge als Befehl erkannt der ist in diesem Kontext nicht zulässig. Man kann derartige Fehler vermeiden, indem Konstantendefinitionen konsequent in Anführungszeichen gesetzt werden.

Als Wert einer Konstanten sind nur skalare Datentypen zulässig, also Zahlen, Zeichenketten und die Booleschen Werte.



Konstanten sind überall im Skript gültig und müssen nicht mit Hilfe der Anweisung global sichtbar gemacht werden.

Fehlerquellen

Gültigkeitsbereich

Vordefinierte Konstanten

PHP kennt bereits einige definierte Konstanten. PHP_VERSION enthält die Versionsnummer des PHP-Interpreters. Sie können damit Skripte schreiben, die sich an bestimmte Bedingungen der Versionen anpassen. PHP_OS berücksichtigt das Betriebssystem, was vor allem bei Operationen im Dateisystem von Bedeutung sein kann.

PHP_VERSION
PHP_OS

In den meisten Programmiersprachen kann der Zustand »Wahr« und »Falsch« durch eine Konstante repräsentiert werden. Aufgrund des schwachen Typkonzepts von PHP wird ersatzweise angenommen, Wert ungleich 0 oder »0« sind »Wahr«. Um trotzdem lesbare Skripte erzeugen zu können, werden die Konstanten TRUE und FALSE verwendet, die intern als 1 und 0 dargestellt werden. Auch die kleine Schreibweise true und false ist zulässig. Ab PHP 4 gibt es auch die Konstante NULL (null), die einen nicht vorhandenen Wert darstellt.

TRUE
FALSE
NULL

Für die Fehlersuche sind zwei Konstanten wichtig, die immer die Datei (Skript) und die Zeilennummer enthalten, die gerade abgearbeitet wird. Haben Sie einen Fehler abgefangen, können Sie in einer Fehler-

FILE
LINE

ausgabe auf diese Konstanten verweisen und so die Quelle des Fehlers feststellen:

- `__FILE__` enthält den Dateinamen des Skripts.
- `__LINE__` die Zeilennummer.

Beachten Sie, dass es sich um zwei Unterstriche vor und nach dem Bezeichner handelt.

E_ERROR
E_WARNING
E_PARSE
E_NOTICE
E_ALL

Bei der Entwicklung von fehlertoleranten Anwendungen sind möglicherweise noch zusätzliche Konstanten von Bedeutung. Die folgenden Konstanten steuern die Ausgabe der Fehlermeldungen zur Laufzeit des Skripts:

- `E_ERROR`, `E_WARNING`, `E_PARSE`, `E_NOTICE`

`E_ALL` bezeichnet alle Fehlertypen. Die Konstanten werden zusammen mit der Funktion `error_reporting` eingesetzt. Mehr dazu finden Sie in ➔Abschnitt 3.5 *Fehlerbehandlung* ab Seite 271. Dort finden Sie auch weitere spezielle Konstanten.

3.2.6 Operatoren

Viele Operatoren haben Sie bei den vorangegangenen Beispielen bereits unbewusst verwendet. Die vielfältigen Schreibweisen sind jedoch eine nähere Betrachtung wert.

Arithmetische Operatoren

+ - * / %

PHP kennt die elementaren arithmetischen Operatoren:

```
$x + $y;    // Addition
$x - $y;    // Subtraktion
$x * $y;    // Multiplikation
$x / $y;    // Division
$x % $y;    // Modulus (Rest der Ganzzahldivision)
```

Bei der Division wird immer dann eine Fließkommadivision durchgeführt, wenn einer der beiden Werte vom Typ `double` ist. Für eine Ganzzahldivision müssen beide Operanden `integer` sein.

Um Werte um eins erhöhen oder verringern zu können, verwenden Sie die Inkrement- und Dekrementoperatoren:

++
--

```
$zahl++
$zahl--
```

Im Zusammenhang mit Zuweisungen ist interessant, ob Sie die Erhöhung (Verringerung) vor oder nach der Zuweisung vornehmen. Entsprechend schreiben Sie den Operator vor oder hinter die Zahl:

```
$x = $y++ // x wird y, dann wird y erhöht
$x = ++$y // y wird erhöht und dann x zugewiesen
$x = $y-- // x wird y, dann wird y verringert
$x = --$y // y wird verringert und dann x zugewiesen
```

Zuweisungsoperatoren

Der einfachste Operator ist der Zuweisungsoperator, der beispielsweise für die Übertragung von Werten in eine Variable Verwendung findet. Sie können die grundlegenden arithmetischen Operatoren mit diesem Operator verbinden:

Zuweisungen

```
=
+=
-=
/=
% =
```

```
$zahl = 45; // weist einer Variablen einen Wert zu
$zahl = $andere_zahl;
```

Das sieht sehr einfach aus. Sie können aber mit Hilfe von Klammern komplexere Konstruktionen schaffen:

```
$zahl = ($faktor = 2) * 4;
$z1 = $z2 = $z3 = $z4 = $z5 = 0;
```

Anschließend enthält die Variable *\$faktor* den Wert 2, *\$zahl* den Wert 8 und alle Variablen *\$zX* enthalten den Wert 0. Die arithmetischen Operatoren können damit kombiniert werden:

```
$zahl += $zahl2;
$zahl -= $zahl2;
$zahl *= $zahl2;
$zahl /= $zahl2;
$zahl %= $zahl2;
```

Auch der Zeichenkettenoperator `.` kann mit der Zuweisung kombiniert werden:

```
$zeichen .= "<br>;
```

Bitoperatoren

Wenn Variablen Werte enthalten, die sich in Byte- oder Bitform auffassen lassen, können Manipulationen mit Bitoperatoren sinnvoll sein. Denken Sie daran, dass ein spezieller binärer Datentyp nicht existiert.

Der Operator `&` führt eine binäre UND-Verknüpfung durch, `|` steht für eine ODER-Verknüpfung, während `~` den Bitwert negiert. Die Operatoren entsprechen Boolescher Algebra.

```
&
|
~
```

Operand \$x	Operand \$y	\$x & \$y	\$x \$y	~\$x	~\$y
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	1

Tabelle 3.5:
Verhalten der
Bitoperatoren

Operand \$x	Operand \$y	\$x & \$y	\$x \$y	~\$x	~\$y
1	1	1	1	0	0

Bedenken Sie, dass das Ergebnis alle Bitstellen beinhaltet, also nicht unbedingt 0 oder 1 ist. Tabelle 3.5 gibt jedoch zur Veranschaulichung nur eine Bitstelle wieder.

>>
<<

Zwei weitere Operatoren arbeiten auf Bit-Ebene: >> und <<. Beide sind mit dem Zuweisungsoperator = verknüpfbar. Mit >> werden Bits nach rechts verschoben, bei << nach links. Bedenken Sie, dass Sie Bitwerte verarbeiten. Es ist also nicht möglich, Variablen der Art "0010" zu verarbeiten:

```
$bit = "0010";
$result = $bit >> 1;
```

Dies ergibt 5, binär 0111. Diese Zeichenkette würde aber dezimal 2 entsprechen – das Ergebnis ist falsch. Versuchen Sie es nun folgendermaßen:

```
$bit = 2;
$result = $bit >> 1;
```

Jetzt wird erwartungsgemäß 1 zurückgegeben. Einmal bitweise nach rechts verschieben entspricht einer Division durch 2, entsprechend ist eine Linksverschiebung eine Multiplikation mit 2.

Logische Operatoren

Logische Operatoren dienen der Konstruktion logischer Ausdrücke und werden im ➡ Abschnitt 3.3.2 *Bedingungen* ab Seite 231 behandelt.

Tabelle 3.6:
Rangfolge der
Operatoren

Assoziativität	Operator
links	,
links	or
links	xor
links	and
rechts	print
links	= += -= *= /= .= %= &= != ~= <<= >>= ^= ~=
links	? :
links	
links	&&
links	
links	^

Assoziativität	Operator
links	&
Nicht assoziativ	== != === !==
Nicht assoziativ	< <= > >=
links	<< >>
links	+ - .
links	* / %
rechts	! ~ ++ -- (int) (double) (string) (array) (object) @
rechts	[
Nicht assoziativ	new

3.2.7 Arrays

Arrays sind eines der vielseitigsten Formen interner Datenspeicherung in fast allen Programmier- und Skriptsprachen. Kaum ein Skript kommt ohne aus und kaum eine Sprache bietet eine so umfangreiche Unterstützung wie PHP.

Was sind Arrays?

Beim Umgang mit Daten werden Sie oft nicht mit einer einzigen Variablen umgehen, sondern mit ganzen Variablensammlungen. Stellen Sie sich vor, Sie würden die Namen der Mitarbeiter einer Firma in Variablen speichern. Arrays bilden solche Variablensammlungen ab. Ein Array enthält also viele Variablenwerte. Dabei sind Sie bei PHP nicht darauf beschränkt, dass diese vom gleichen Typ sind – jedes Element kann jeden Datentyp haben. Auch Arrays sind als Elemente eines Arrays zulässig.

Arrays besitzen Dimensionen. Das einfachste Array hat nur eine Dimension. Es kann beliebig viele Werte aufnehmen, die man auch als Elemente bezeichnet. Diese Form kann man sich gut als Tabelle vorstellen, die aus nur einer Spalte und beliebig vielen Zeilen besteht. Die voraussichtliche Größe müssen Sie nicht bestimmen, PHP wird die Grenzen des Arrays automatisch neu festlegen, wenn weitere Werte zugewiesen werden. Im Gegensatz zu Datenbanken oder Excel-Tabellen haben die Spalten jedoch keine Namen oder Nummern – jedes Element wird beim Lesen direkt angesprochen. Beim Schreiben kann die Wahl der Zeile aber auch PHP überlassen werden.

Um die Namen der Mitarbeiter zu speichern, nutzen Sie die folgende Schreibweise einer Variablenzuweisung: **Indizierte Arrays**

```
<?php
$mitarbeiter[] = "Clemens Krause"
$mitarbeiter[] = "Janine Bünning"
$mitarbeiter[] = "Carolin Scholz"
$mitarbeiter[] = "Carolin Schröder"
?>
```

Daraus entsteht ein eindimensionales Array. Dieses Array ist intern indiziert. Ein Index ist ein Verweis auf ein Element. Arrays werden mit 0 beginnend indiziert. Sie könnten auch gleich die Indexwerte angeben. Das folgende Beispiel ist äquivalent dem ersten:

```
<?php
$mitarbeiter[0] = "Clemens Krause";
$mitarbeiter[1] = "Janine Bünning";
$mitarbeiter[2] = "Carolin Scholz";
$mitarbeiter[3] = "Carolin Schröder";
?>
```

Die Angabe muss aber nicht in der Reihenfolge und auch nicht lückenlos erfolgen. Wenn Sie ein Element auf einen bestimmten Index setzen, setzt PHP danach mit der nächsten Ganzzahl fort, falls keine Angabe erfolgt. Im folgenden Beispiel erhalten die Mitarbeiter die Indizes 5 bis 7 und 12:

```
<?php
$mitarbeiter[5] = "Clemens Krause";
$mitarbeiter[] = "Janine Bünning";
$mitarbeiter[] = "Carolin Scholz";
$mitarbeiter[12] = "Carolin Schröder";
?>
```

array()

Sie können alternativ auch das Schlüsselwort `array` verwenden, mit dem sich ebenso Werte zuweisen lassen. Arrays können verschachtelt werden, um Arrays von Arrays von Arrays usw. zu erzeugen. Damit lassen sich auch sehr komplexe Datenstrukturen abbilden. Dies kann man zwar auch mit der Klammerdarstellung machen, die Verwendung der Anweisung `array` ist jedoch meist einfacher.

*Listing 3.13:
array_simple:
Einfaches, indiziertes
Array*

```
<?php
$mitarbeiter = array("Clemens Krause",
                    "Janine Bünning",
                    "Carolin Scholz",
                    "Carolin Schröder");
echo "Mitarbeiter 1: $mitarbeiter[0]<br>";
echo "Mitarbeiter 2: $mitarbeiter[1]<br>";
echo "Mitarbeiter 3: $mitarbeiter[2]<br>";
echo "Mitarbeiter 4: $mitarbeiter[3]";
?>
```

Beachten Sie, dass aus Darstellungsgründen große Arrays oft über mehrere Zeilen geschrieben werden. Dies dient nur der Übersichtlichkeit der Skripte. Dem PHP-Interpreter ist die Schreibweise egal.



Oft werden Sie mit Schlüssel-/Wertepaaren arbeiten. Auch dafür werden Arrays eingesetzt. Sie können statt der fortlaufenden Indizes Schlüssel, beispielsweise Zeichenketten, als Merkmal angeben. Solche Arrays werden assoziative Arrays genannt. Das bereits gezeigte Beispiel könnte auch folgendermaßen geschrieben werden:

Assoziative Arrays

```
<?php
$mitarbeiter["M1"] = "Clemens Krause";
$mitarbeiter["M2"] = "Janine Bünning";
$mitarbeiter["M3"] = "Carolin Scholz";
$mitarbeiter["M4"] = "Carolin Schröder";
?>
```

Erzeugen Sie ein leeres Array mit der Anweisung `array`, um sicher zu stellen, dass der Interpreter dies erkennt. Das ist im Sinne eines guten Programmierstils sinnvoll, denn falls weitere Zuweisungen misslingen, hat die Variable wenigstens den richtigen Datentyp. Die Funktion `is_array` gibt dann `TRUE` zurück. Die Anzahl der Elemente (mit `count` ermittelt) ist 0.

Leeres Array

Dann weisen Sie die Werte zu, wie im folgenden Beispiel gezeigt:

```
<?php
$myarray = array();
$myarray[] = "Erster Wert";
$myarray[] = "Zweiter Wert";
?>
```

Wenn Sie die Anweisung `array` einsetzen, um assoziative Arrays zu erzeugen, benötigen Sie zur Angabe der Schlüsselwerte den zusätzlichen Operator `=>`.

Der Operator =>

```
<?php
$mitarbeiter = array("M1" => "Clemens Krause",
                    "M2" => "Janine Bünning",
                    "M3" => "Carolin Scholz",
                    "M4" => "Carolin Schröder");
echo "Mitarbeiter 1: " . $mitarbeiter["M1"] . "<br>";
echo "Mitarbeiter 2: " . $mitarbeiter["M2"] . "<br>";
echo "Mitarbeiter 3: " . $mitarbeiter["M3"] . "<br>";
echo "Mitarbeiter 4: " . $mitarbeiter["M4"];
?>
```

Listing 3.14:
array associative:
Assoziatives Array

Dies bringt zwar keinen Leistungsgewinn, der Quelltext wird aber bei konsequenter Anwendung leichter lesbar. Echte Vorteile ergeben sich bei der Nutzung für Daten, deren Umfang und Struktur Sie nicht genau kennen. Wenn Sie indizierte Arrays mit Zählschleifen abfragen, kommen nur numerische Indizes in Betracht. Der direkte Zugriff auf

ein bestimmtes Element, dessen Index Sie nicht kennen, bleibt Ihnen verwehrt. Allerdings gibt es einige Funktionen, die beim schrittweisen Durchlaufen numerischer Arrays helfen. Dazu finden Sie mehr im ➔ Abschnitt *Arrayfunktionen* ab der nächsten Seite.

Arrays und Arrayelemente löschen

Wenn Sie einem Array eine leere Zeichenkette zuweisen, wird es nicht gelöscht. Dabei geht zwar die innere Struktur verloren, die ist aber sowieso bedeutungslos, denn sie muss nicht vor der Verwendung deklariert werden. Um ein Array völlig zu löschen, verwenden Sie `unset`:

```
unset($arr);
```

Array leeren

Mit `array()` kann ein Array auch geleert werden. Ist eine Funktion des Skripts auf den Datentyp `array` angewiesen, führt das zu Fehlverhalten:

```
$arr = array();
```

Ein einzelnes Element eines Array entfernen Sie ebenso mit `unset`:

```
unset($arr[14]);
```

Mehrdimensionale Arrays

`$arr[0][1]`

Arrays können mehrere Dimensionen haben. Im Beispiel wäre eine Trennung von Vor- und Zunamen sinnvoll. Dazu nutzen Sie einfach den `array`-Befehl innerhalb eines übergeordneten `array`-Befehls:

*Listing 3.15:
array_multi:
Mehrdimensionale
Arrays verwenden*

```
$mitarbeiter = array("M1" => array("Clemens", "Krause"),
                    "M2" => array("Janine", "Bünning"),
                    "M3" => array("Carolin", "Scholz"),
                    "M4" => array("Carolin", "Schröder")
                    );
echo "Mitarbeiter 1: ", $mitarbeiter["M1"][0], " ",
    $mitarbeiter["M1"][1], "<br>";
echo "Mitarbeiter 2: ", $mitarbeiter["M2"][0], " ",
    $mitarbeiter["M2"][1], "<br>";
echo "Mitarbeiter 3: ", $mitarbeiter["M3"][0], " ",
    $mitarbeiter["M3"][1], "<br>";
echo "Mitarbeiter 4: ", $mitarbeiter["M4"][0], " ",
    $mitarbeiter["M4"][1], "<br>";
```

Beachten Sie die Schreibweise beim Abruf der Inhalte des Arrays. Die im assoziativen Array verwendeten Schlüssel werden hier *ohne* Anführungszeichen geschrieben. Weitere Dimensionen werden einfach dahinter in eckige Klammern gesetzt.

Das Beispiel zeigt auch zugleich, wie indizierte und assoziative Arrays gemischt werden können. Wegen der mangelnden Übersichtlichkeit sollten Sie davon nur sehr vorsichtig Gebrauch machen.

Wenn Sie an das so erzeugte Array weitere Einträge anhängen, liegt die folgende Schreibweise nahe: **Fehlerquellen**

```
$mitarbeiter = array("M5" => array("Vor", "Nach");
```

Das wird nicht funktionieren. Sie würden nämlich das Array überschreiben. Sie können ein vorhandenes zweidimensionales Array leicht erweitern, indem der neue Schlüssel als assoziativer Index genutzt wird:

```
$mitarbeiter["M5"] = array("Vor", "Nach");
```

Arrayfunktionen

Arrays werden in PHP durch eine ganze Palette von Funktionen unterstützt. Diese Funktionen werden nachfolgend ausführlich behandelt, da sich damit viele alltägliche Probleme effizient lösen lassen. Vor der ersten Verwendung sollten Sie jedoch das beschriebene Anlegen und Abrufen von Arrays »von Hand« beherrschen.

Folgende Funktionen stehen zur Verfügung:

- `each`
Liefert alle Elemente nacheinander.
- `list`
Überführt die Elemente in skalare Variablen.
- `count, sizeof`
Liefert die Anzahl der Elemente eines Arrays.
- `current, pos`
Aktuelle Position des internen Array-Zeigers.
- `key`
Liefert einen Schlüsselwert.
- `end`
Setzt den internen Zeiger ans Ende.
- `next`
Setzt den internen Zeiger eins weiter.
- `rev`
Setzt den internen Zeiger eins zurück.

**Funktionen für
mehrere Elemente**

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

- reset

Setzt den internen Zeiger an den Anfang zurück.

Einige Funktionen dienen speziell zum Sortieren von Elementen:

Sortierfunktionen

- nat sort

Sortiert ein Array natürlich, also »2« steht vor »10«. Normalerweise sortieren Computer zeichenweisen, dann steht »10« vor »2«, weil zuerst die erste Stelle verglichen wird.

- arsort

Sortiert ein Array rückwärts und behält die Indexzuordnung bei.

- asort

Sortiert ein Array unter Beibehaltung der Indexzuordnung.

- ksort

Sortiert ein Array anhand der Schlüsselwerte.

- rsort

Sortiert ein eindimensionales Array rückwärts.

- sort

Sortiert ein eindimensionales Array vorwärts.

- uasort

Diese Funktion sortiert ein assoziatives Array unter Nutzung einer selbstdefinierten Sortierfunktion und behält dabei die Indexzuordnung bei.

- uksort

Diese Funktion sortiert ein Array unter Nutzung einer selbstdefinierten Sortierfunktion anhand der Schlüsselwerte.

- usort

Diese Funktion sortiert ein eindimensionales Array unter Nutzung einer selbstdefinierten Sortierfunktion.

Diese Funktionen lösen spezielle, seltener auftretende Probleme:

Sonstige Funktionen

- range

Gibt ein Array aus Ganzzahlen zurück, die zwischen einem Anfangs- und einem Endwert liegen.

- shuffle

Bringt ein Array »durcheinander«.

In PHP 4 kamen eine ganze Reihe von Funktionen hinzu, auf die vor allem Umsteiger von Perl gewartet haben. Diese basieren nicht auf der internen Zeigerverwaltung und realisieren komplexere Operationen mit Arrays:

- `array_count_values`
Erstellt ein Array, das die Elemente des übergebenen Arrays als Schlüssel und die Häufigkeit des Auftretens als Wert enthält.
- `array_keys`
Gibt die Schlüssel eines zweidimensionalen Arrays als eindimensionales Array zurück.
- `array_filter`
Filtert die Elemente eines Arrays unter Verwendung einer selbstdefinierten Filterfunktion.
- `array_flip`
Diese Funktion vertauscht Schlüssel und Wert in einem assoziativen Array. Ein Anwendungsbeispiel finden Sie in ➡ Abschnitt 3.6 *Hilfsfunktionen* ab Seite 294.
- `array_map`
Verknüpft mehrere Arrays mit Hilfe einer selbstdefinierten Funktion.
- `array_merge`
Diese Funktion verbindet zwei oder mehrere Arrays.
- `array_merge_recursive`
Verbindet Arrays rekursiv, wobei gleiche Schlüssel zusammengeführt werden.
- `array_multisort`
Sortiert mehrere oder mehrdimensionale Arrays.
- `array_intersect`
Berechnet die Durchschnittsmenge mehrere Arrays und gibt damit die gemeinsamen Elemente zurück.
- `array_pad`
Vergrößert ein Array am Anfang oder am Ende. Dabei werden die neuen Elemente mit einem Standardwert aufgefüllt.

- `array_pop`
Entfernt das letzte Element eines Arrays und gibt es zurück. Das Array ist danach ein Element kleiner.
- `array_push`
Fügt ein Element am Ende eines Arrays hinzu. Das Array wächst dabei um ein Element.
- `array_rand`
Gibt zufallsgesteuert ein Element eines Arrays zurück.
- `array_reduce`
Diese Funktion verarbeitet die Elemente eines Arrays zu einem skalaren Wert. Dabei muss eine eigene »Reduktionsfunktion« zur Verfügung gestellt werden.
- `array_reverse`
Dreht ein Array um. Das letzte Element steht nun an der ersten Position und umgekehrt.
- `array_search`
Durchsucht ein Array und gibt die Indizes der Fundstellen zurück.
- `array_shift`
Entfernt ein Element vom Anfang und gibt es zurück. Alle Elemente rutschen eine Position nach unten. Das Array ist nun um ein Element kleiner.
- `array_slice`
Gibt einige (bestimmte) Elemente des Arrays zurück. Die Elemente werden anhand der Position im Array ausgewählt.
- `array_splice`
Ersetzt einige (bestimmte) Elemente des Arrays. Die durch die Position im Array beschriebenen Elemente werden durch bestimmte Werte ersetzt. Kann auch verwendet werden, um einen Teil eines Arrays zu entfernen.
- `array_unshift`
Fügt mehrere Elemente am Anfang eines Arrays ein. Die Elemente werden einzeln, nicht als Array übergeben. Das resultierende Array wächst dabei um mehrere Elemente.

- `array_unique`
Entfernt doppelte Elemente in einem Array.
- `array_values`
Liefert ein eindimensionales Array mit den Werten eines assoziativen Arrays zurück.
- `array_walk`
Wendet eine Funktion auf alle Elemente an.
- `compact`
Überführt mehrere skalare Variablen in ein Array. Als Parameter werden die Namen der Variablen angegeben.
- `extract`
Erstellt aus einem assoziativen Array Variablen. Dabei ergeben die Schlüssel die Namen und die Werte den Inhalt der Variablen.
- `in_array`
Untersucht ein Array auf das Vorhandensein von Elementen mit bestimmten Eigenschaften und gibt einen Booleschen Wert zurück.

Grundsätzliches

Umsteiger aus anderen Skriptsprachen wie VBScript sollten sich ebenso intensiv mit den Array-Funktionen beschäftigen. PHP bietet hier einiges mehr und erlaubt manche elegante Lösung. Perl-Programmierer wurden bereits bei der Vorstellung der PHP 4-Funktionen angesprochen. Hier bot PHP 3 leider nicht den Funktionsumfang, der in der Praxis benötigt wird.

Unabhängig von der strengen Reihenfolge nach Funktionsklassen sollten Sie auch die Abschnitte zu Zeichenketten (beginnend mit ➡ Abschnitt 3.2.8 *Zeichenkettenverarbeitung* ab Seite 189) lesen. Sie finden dort Funktionen, die bei Problemen mit Arrays oft naheliegende Lösungen erlauben. Beide Abschnitte sind unabhängig voneinander und ergänzen sich gegenseitig in hervorragender Weise.

Aus naheliegenden (nämlich praktischen) Gründen nutzen einige der folgenden Beispiele bereits Kontrollbefehle zur Bildung von Schleifen oder Funktionen. Lesen Sie die entsprechenden Abschnitte weiter hinten in diesem Kapitel (beginnend mit dem ➡ Abschnitt 3.3 *Programmieren mit PHP* ab Seite 230), wenn Sie Probleme haben, die Skripte zu lesen. Für das grundsätzliche Verständnis der Array-Funktionen ist das jedoch meist nicht notwendig.

**Wo Sie noch
Informationen
finden**

Kontrollausgaben Manchmal werden Sie den gesamten Inhalt eines Arrays betrachten wollen – beispielsweise zu Testzwecken. Dann ist `print_r` die passende Funktion, der das Array übergeben wird. Dies ist eine Hilfsfunktion zur Fehlersuche, die jede Variable komplett ausgibt.

Listing 3.16:
array_print_r:
Arrays ausgeben

```
<?php
$mitarbeiter = array("M1"=>array("Clemens", "Krause"),
                    "M2"=>array("Janine", "Bünning"),
                    "M3"=>array("Carolin", "Scholz"),
                    "M4"=>array("Carolin", "Schröder") );
print_r($mitarbeiter);
?>
```

Funktionen für mehrere Elemente

array_walk()

Oft werden Funktionen genutzt, um Elemente zu bearbeiten. Um nun eine Funktion, egal ob intern oder selbst definiert, auf alle Elemente eines Arrays anzuwenden, verwenden Sie die Funktion `array_walk`.

Listing 3.17:
array_walk: Arrays
mit eigener Funktion
verarbeiten

```
<?php
function print_array($element) {
    echo "Mitarbeiter: $element[0] $element[1] <br>";
}
$mitarbeiter = array("M1"=>array("Clemens", "Krause"),
                    "M2"=>array("Janine", "Bünning"),
                    "M3"=>array("Carolin", "Scholz"),
                    "M4"=>array("Carolin", "Schröder") );
$mitarbeiter["M5"] = array("Vorname", "Nachname");
array_walk( $mitarbeiter , "print_array" );
?>
```

Die konkrete Anzahl der Elemente des Array müssen Sie dazu nicht kennen. `array_walk` arbeitet immer mit der ersten Dimension und übergibt jedes Element, egal ob Array oder nicht, an die aufgerufene Funktion. Theoretisch kann man hier erneut mit `array_walk` arbeiten, um die nächste Dimension aufzulösen.



Hinweis

Wenn die Funktion, die `array_walk` aufruft, nicht gefunden wird, passiert nichts – es wird auch keine Fehlermeldung ausgegeben. Achten Sie auf korrekte Schreibweise und testen Sie die Funktion separat aus.

list()

Mit Hilfe von `list` übertragen Sie die Elemente eines Arrays auf einzelne Variablen. Dies erleichtert unter Umständen die Weiterverarbeitung. So können Sie die Funktion `print_array` aus Listing 3.18 auch folgendermaßen schreiben:

Listing 3.18:
array_list:
Anwendung von
list

```
<?php
function print_array($element) {
    list($vorname, $nachname) = $element;
    echo "Mitarbeiter: $vorname $nachname <br>";
}
?>
```

Bei der praktischen Arbeit mit Arrays werden sicher nicht immer alle Werte durchlaufen. PHP verfügt über einige Funktionen, mit denen gezielt Elemente ausgewählt werden können. Intern arbeiten Arrays mit einem Zeiger, der auf ein bestimmtes – das so genannte aktuelle – Element verweist.

Die Funktion `each` stellt praktisch die ideale Ergänzung zu `list` dar. Hier wird ein komplettes Schlüssel/-Wertepaar an ein eindimensionales Array übergeben. Genauer werden vier Werte übergeben, die den Schlüsseln 0, 1, key und value entsprechen. Die Anwendung ist vor allem in Schleifen sinnvoll. Sind keine Werte mehr vorhanden, gibt die Funktion `FALSE` zurück und die Schleife endet, wie es das folgende Beispiel zeigt:

```
<?php
while ($ma = each($mitarbeiter)) {
    echo "0: ", $ma["0"], " - ";
    echo "1: ", $ma["1"], ", ";
    echo "key: ", $ma["key"], " - ";
    echo "value: ", $ma["value"], "<br>";
}
?>
```

Listing 3.19:
array_each:
Anwendung von
`each`

Dies ist übrigens eine Eigenschaft aller assoziativen Arrays. Die numerischen Schlüssel und die mit den Namen `key` und `value` werden nicht von `each` produziert, sondern stehen generell zur Verfügung.



Hinweis

In Kombination mit `list` lassen sich so auch komplexe Arrays sehr leicht in ihre Bestandteile zerlegen, die Ausgabe von `each` wird dabei direkt an `list` übergeben, wie das nächste Beispiel zeigt. Sehr häufig wird der gesamte Ausdruck in einer `while`-Schleife eingesetzt. Findet `each` keine weiteren Elemente, wird `FALSE` zurückgegeben und die Schleife beendet. Das Weitersetzen des internen Zeigers erledigt `each` dabei automatisch.

**each und list
kombinieren**

```
<?php
$mitarbeiter = array("M1" => array("Clemens", "Krause"),
                    "M2" => array("Janine", "Bünning"),
                    "M3" => array("Carolin", "Scholz"),
                    "M4" => array("Carolin", "Schröder"));
list(, $ma) = each( $mitarbeiter );
list( $vn, $nn ) = $ma;
echo "Vorname: $vn <br>";
echo "Nachname: $nn <br>";
?>
```

Listing 3.20:
array_eachlist: `each`
in Kombination mit
`list`

Wenn Sie `each` mehrfach anwenden, werden nacheinander alle Elemente des Array ausgegeben. Der Zeiger, der auf das aktuelle Element des Arrays zeigt, wandert im Array weiter. Dies ist unter Umständen nicht erwünscht. Sie können deshalb mit den Funktionen `prev` und `next` den Zeiger vor und zurück setzen. Das folgende Listing zeigt den

**prev()
next()**

entsprechenden Ausschnitt. Außerdem finden Sie die Syntax für `list` unter Fortlassung des ersten Parameters:

Listing 3.21:
array_prevnext:
next, prev und list

```
<?php
$mitarbeiter = array("M1" => "Clemens Krause",
                    "M2" => "Janine Bünning",
                    "M3" => "Carolin Scholz",
                    "M4" => "Carolin Schröder");

list($name1) = each($mitarbeiter);
next($mitarbeiter);
list($name2) = each($mitarbeiter);
echo "Name: ", $name1 , "<br>";
echo "Name: ", $name2 , "<br>";
/* prev */
end($mitarbeiter);
prev($mitarbeiter);
list($name1) = each($mitarbeiter);
list($name2) = each($mitarbeiter);
echo "Name: ", $name1 , "<br>";
echo "Name: ", $name2 , "<br>";
?>
```

reset()
end()

Um einen definierten Ausgangspunkt für den Cursor zu bekommen, verwenden Sie `reset` und `end`:

```
reset( $mitarbeiter );
end( $mitarbeiter );
```

Mit `reset` wird der Cursor auf den ersten Eintrag des Arrays gesetzt. Den letzten Eintrag erreichen Sie mit `end`.

current()
pos()

Das automatische Weitersetzen des Cursors mit `each` ist manchmal störend. Sie können dann die Funktion `current` anwenden. Wird ein Array durchlaufen, ist die Anwendung von `next` zwingend.

Listing 3.22:
array_current:
Anwendung von
current(Ausschnitt)

```
<?php
$mitarbeiter = array("M1" => "Clemens Krause",
                    "M2" => "Janine Bünning",
                    "M3" => "Carolin Scholz",
                    "M4" => "Carolin Schröder");

$name1 = current($mitarbeiter);
next($mitarbeiter);    # Nächster
$name2 = current($mitarbeiter);
echo "Name: ", $name1 , "<br>";
echo "Name: ", $name2 , "<br>";
$name1 = current($mitarbeiter);
prev($mitarbeiter);    # Vorhergehender
$name2 = current($mitarbeiter);
echo "Name: ", $name1 , "<br>";
echo "Name: ", $name2 , "<br>";
?>
```

Beachten Sie, dass `current` nicht zusätzlich noch Elemente mit den Schlüsseln 0 und 1 wiedergibt, der `list`-Befehl ist deshalb nicht erforderlich (siehe Listing 3.20 für ein eindimensionales Array). In manchen Skripten finden Sie noch den Befehl `pos`, dies ist lediglich eine andere Schreibweise für `current`.

Um den Schlüssel der aktuellen Position des Cursors alleine zu ermitteln, können Sie `key` einsetzen:

```
<?php
$mitarbeiter = array("M1" => "Clemens Krause",
                    "M2" => "Janine Bünning",
                    "M3" => "Carolin Scholz",
                    "M4" => "Carolin Schröder");
$mykey = key($mitarbeiter);
echo "Eintrag $mykey: " . $mitarbeiter[$mykey];
?>
```

Listing 3.23:
array_key: Einsatz
von `key`

Erweiterte Array-Funktionen

Für die Bearbeitung von Arrays stehen in PHP 4 nicht nur neue Funktionen zur Verfügung, sondern auch eine spezielle Schleife, die Standardfälle bedeutend vereinfacht: `foreach`. Im Detail wird diese Konstruktion im ➡ Abschnitt *Arrays mit foreach bearbeiten* ab Seite 244 behandelt. In den folgenden Beispielen werden diese Funktionen zur Ausgabe der Ergebnisse verwendet.

In einigen Beispielen dieses Abschnitts werden folgende Musterdaten verwendet, die jedem Skript vorangestellt werden müssen: **Musterdaten**

```
$ma = array("Name" => "Krause",
           "Ort" => "Berlin",
           "PLZ" => 12683);
```

Das folgende Beispiel zeigt, wie die Schlüssel mit konventionellen Funktionen extrahiert werden:

```
for(reset($ma); $key = key($ma); next($ma)) {
    echo "ma[$key] = ".$ma[$key];
    echo "<br>";
}
```

Einfacher ist die Anwendung der Funktion `array_keys`. Wenn Sie dagegen die Werte benötigen, wird `array_values` genutzt:

array_keys()
array_values()

```
<?php
$mykeys = array_keys($ma);
foreach ($mykeys as $keyname) {
    echo "$keyname<br>";
}
?>
```

Listing 3.24:
array_keys:
Anwendung der
Funktion
`array_keys`

Listing 3.25:
array_values: Die
Funktion
array_values in
Aktion

```
<?php
$myvalues = array_values($ma);
foreach ($myvalues as $value) {
    echo "$values<br>";
}
?>
```

Die erweiterte Syntax der foreach-Schleife kann dies aber auch ohne diese Funktion:

```
foreach($mykeys as $keyname => $value) {
    echo "$keyname<br>";
}
```

Dieses Skript verhält sich genauso wie das in Listing 3.24. Allerdings wäre die zweite Variante, wenn wirklich nur die Schlüssel benötigt werden, unsauberer programmiert. Der Sinn der Sequenz ist nicht völlig klar, was die Funktion `array_keys` im Gegensatz dazu sicherstellen kann.

array_reverse()

Die Funktion `array_reverse` dreht ein Array um. Handelt es sich um ein assoziatives Array, werden alle Elemente umgedreht.

Listing 3.26:
array_reverse:
Arrays mit
array_reverse
umdrehen

```
<?php
$ma2 = array_reverse($ma);
foreach ($ma2 as $keyname => $value) {
    echo "<b>$keyname</b>: $value<br>";
}
?>
```

array_pad()

Für die Weiterverarbeitung von Arrays ist es oft notwendig, dass zwei Arrays die gleiche Länge aufweisen. Hier kommt `array_pad` zum Einsatz. Die Funktion füllt am Anfang oder am Ende ein Array mit Standardwerten auf.

Listing 3.27:
array_pad: Array mit
array_pad auffüllen

```
<?php
$arr1 = array(1, 3, 5, 7, 9);
$int1 = count($arr1);
$arr2 = array(2, 4);
$int2 = count($arr2);
if ($int1 > $int2) {
    $arr2 = array_pad($arr2, $int1, 0);
}
?>
```

Der zweite Parameter bestimmt, wie viele Elemente aufgefüllt werden. Normalerweise werden diese am Ende angefügt. Ist der Wert negativ, werden sie dagegen am Anfang eingefügt.

array_merge()

Statt Standardwerten kann man auch ein anderes Array anhängen. Dazu wird die Funktion `array_merge` verwendet. Welche Konstruktion die Arrays haben, spielt keine Rolle. Die Funktion verkraftet außerdem mehrere Argumente.

```
<?php
$arr1 = array("Name" => "Krause",
              "Ort" => "Berlin",
              "PLZ" => 12683);
$arr2 = array("ISBN-Liste" =>
              array("3-446-21301-5",
                    "3-446-21099-7",
                    "3-446-21098-9",
                    "3-446-19378-2",
                    "3-446-19550-5",
                    "3-446-21076-8",
                    "3-8273-1548-4",
                    "3-8273-1427-5")
             );
$arrTitles = array_merge($arr1, $arr2);
foreach ($arrTitles as $key => $value) {
    echo "<b>$key</b>: ";
    if (is_array($value)) {
        foreach($value as $val) {
            echo "<br>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~";
        }
    } else {
        echo " $value<br>";
    }
}
}>
```

Listing 3.28:
array_merge:
Verknüpfung zweier
unterschiedlich
aufgebauter Arrays


Mit `array_merge` wird das Arrays `$arr2` als weiteres Element angehängt. Die Ausgabe des kombinierten Arrays erfolgt in zwei Schleifen, wobei in der inneren mit `is_array` überprüft wird, ob wirklich ein Array im Array auftritt. Andernfalls wird der skalare Wert ausgegeben.

Wie es funktioniert

Name: Krause
Ort: Berlin
PLZ: 12683
ISBN-Liste:
3-446-21301-5
3-446-21099-7
3-446-21098-9
3-446-19378-2
3-446-19550-5
3-446-21076-8
3-8273-1548-4
3-8273-1427-5

Abbildung 3.1:
Ausgabe aus Listing
3.28

Sie sind also in der Konstruktion der Arrays kaum Beschränkungen unterworfen. Mit geschickten Abfragen lässt sich alles wieder auseinandernehmen, beispielsweise unter Nutzung von `is array`.

Ein Beispiel für die Nutzung von `array_splice` finden Sie im XML-Projekt in  Kapitel 9 *Fortgeschrittene Programmierung* ab Seite 629. Im dritten Praxiskomplex, wo das Templatesystem `PHPTemple` vorgestellt wird, finden Sie weitere Anwendungen der Array-Funktionen mit Erläuterungen.

Sortierfunktionen

**asort()
arsort()**

Zu einer effizienten Datenverarbeitung gehören auch Sortierfunktionen. Hier bietet PHP eine überdurchschnittliche Unterstützung. Das folgende Beispiel sortiert das bereits mehrfach verwendete eindimensionale Mitarbeiter-Array und gibt es aus:

*Listing 3.29:
array_asort:
Anwendung von
asort*

```
<?php
asort($ma);
for(reset($ma); $key = key($ma); next($ma)) {
    echo "ma[$key] = " . $ma[$key] . "<br>";
}
?>
```

Der Funktionsname asort steht dabei für »associative sort«, entsprechend wird mit arsort (»associative reverse sort«) rückwärts sortiert. Dies bedeutet, dass zu den Elementen assoziierte Schlüsselwerte assoziativer Arrays mit verschoben werden.

**sort()
rsort()**

Bei indizierten Arrays verwenden Sie sort und rsort entsprechend. Alle Sortierbefehle sortieren alphabetisch.

ksort()

Wenn Sie das Array wieder in die ursprüngliche Reihenfolge bringen möchten, wenden Sie ksort an; die Funktion sortiert aufsteigend nach den Schlüsseln:

```
ksort($mitarbeiter);
```

**usort()
ursort()
uksort()**

Wenn Sie nicht nur alphabetisch sortieren möchten, können Sie die Funktionen usort, ursort und uksort anwenden. Gegenüber den bereits beschriebenen Variationen können Sie als zusätzlichen Parameter eine Funktion übergeben, die bestimmte Sortiermerkmale bestimmt. Die Funktion muss den Sortiervorgang nach folgendem Schema – durch Rückgabe eines bestimmten Codes – steuern:

- 0. Zwei miteinander verglichene Werte sind gleich.
- 1. Der zweite Parameter ist größer als der erste.
- -1. Der erste Parameter ist größer als der zweite.

Entsprechend interpretiert PHP beim Sortiervorgang dies als Einordnungskriterium. Eine Anwendung zeigt das folgende Skript, welches Einträge in einem Array nach deren Länge sortiert:

*Listing 3.30:
array_usort: Eine
eigene Sortier-
funktion mit usort*

```
<?php
function lencmp($a,$b) {
    if (strlen($a) == strlen($b)) return 0;
    return (strlen($a) > strlen($b)) ? -1 : 1;
}
$testarray = array("M1" => "Längstes Wort",
                   "M2" => "Kurz",
                   "M3" => "Läääänger",
                   "M4" => "Weiß nicht");
```

```
usort($testarray, lncmp);
while(list($key,$value) = each($testarray)) {
    echo "$key: $value (" . strlen($value) . ")<br>";
}
?>
```

Alle Sortierfunktionen verändern das als Parameter übergebene Array direkt. Zurückgegeben wird im Erfolgsfall TRUE, sonst FALSE.

3.2.8 Zeichenkettenverarbeitung

Die umfangreichen Möglichkeiten des Umgangs mit Arrays deuten an, dass PHP generell eine hervorragende Unterstützung für Zeichenketten besitzt. Tatsächlich gibt es fast für jeden Zweck eine spezielle Funktion.

Übersicht Zeichenkettenfunktionen

Hier eine alphabetische Liste der Zeichenkettenfunktionen:

- addslashes
Setzt ein Backslash \ vor Sonderzeichen.
- addcslashes
Fügt C-typische Escape-Zeichen hinzu.
- stripslashes
Entfernt C-typische Escape-Zeichen.
- chr
Gibt ein Zeichen für einen ASCII-Wert zurück.
- ord
Gibt den ASCII-Wert eines Zeichens zurück.

Drei Funktionen helfen beim Verschlüsseln bzw. codieren:

- crc32
32 Bit-Polynom (Prüfsumme) einer Zeichenkette
- crypt
Verschlüsselt eine Zeichenkette nach DES.
- md5
Berechnet die MD5-Zeichenfolge.

Besonders für die Programmierung von Webseiten sind viele Zeichenkettenfunktionen vorhanden:

**Funktionen für
spezielle Zeichen
und Zeichen-
funktionen**

**Verschlüsselungs-
funktionen**

HTML- und Web-Funktionen

- `flush`
Leert den Ausgabepuffer.
- `get_meta_tags`
Extrahiert <META>-Tags aus einer Datei.
- `htmlspecialchars`
Wandelt in spezielle HTML-Symbole um.
- `htmlentities`
Wandelt in spezielle HTML-Symbole um.
- `nl2br`
Wandelt native Zeilenumbrüche in
-Tags.
- `quoted_printable_decode`
Dekodiert Zeichenfolgen, die mit »Quoted Printable« kodiert wurden.
- `quotemeta`
Setzt ein Backslash vor jedes Sonderzeichen.
- `rawurldecode`
Macht eine Zeichenfolge URL-fähig.
- `rawurlencode`
Wandelt eine URL in eine Zeichenfolge um.
- `parse_str`
Analysiert den QueryString.
- `get_html_translation_table`
Gibt ein Array zurück, das die Übersetzungstabelle für HTML-Zeichen enthält.
- `strip_tags`
Entfernt HTML- und PHP-Tags.

Einige Funktionen dienen der Ausgabeformatierung:

Aus- und Eingabefunktionen

- `print`
Gibt Argumente aus (siehe zum Vergleich auch bei `echo`).
- `printf`
Gibt Argumente formatiert aus.

- `sprintf`
Gibt eine formatierte Zeichenkette zurück.
- `setlocale`
Setzt eine sprachliche Lokalisierung.
- `sscanf`
Erkennt formatierte Zeichen in Eingabezeichenfolgen.
- `localeconv`
Informationen über die Formatierung.

Mit den folgenden Funktionen werden Arrays verarbeitet:

- `implode`
Fasst ein Array zu einer Zeichenkette zusammen.
- `join`
Dies ist ein Alias für `implode`.
- `explode`
Teilt eine Zeichenkette an einem Trennzeichen und gibt die Fragmente als Array zurück.

Array-spezifische Funktionen

Die Verarbeitung von Zeichenketten erledigen die folgenden Funktionen auf vielfältige Weise:

- `chop`
Entfernt Leerzeichen am Ende der Zeichenkette.
- `chunk_split`
Teilt eine Zeichenkette nach Bytes.
- `convert_cyr_string`
Konvertiert kyrillische Zeichensätze.
- `ltrim`
Entfernt Leerzeichen am linken Ende.
- `strcasecmp`
Vergleicht zwei Zeichenketten ohne Berücksichtigung von Groß- und Kleinschreibung.
- `strchr`
Findet das Auftreten eines Zeichens in einer Zeichenkette.

- `strcmp`
Vergleicht zwei Zeichenketten.
- `strcoll`
Zeichenkettenvergleich auf Basis sprachspezifischer Informationen.
- `strcspn`
Gibt einen Teil einer Zeichenkette zurück, die nicht Zeichen einer zweiten Zeichenkette enthält.
- `stripslashes`
Entfernt zur Markierung von Sonderzeichen verwendete Backslashes.
- `strlen`
Gibt die Länge einer Zeichenkette zurück.
- `strrpos`
Sucht ein Zeichen vom Ende der Zeichenkette an.
- `strpos`
Sucht ein Zeichen vom Anfang der Zeichenkette an.
- `strchr`
Wie `strchr`, aber vom Ende beginnend.
- `strrev`
Dreht eine Zeichenkette um.
- `strspn`
Erkennt Zeichen in einer Zeichenkette und gibt diesen Teil zurück.
- `strstr`
Entspricht `strchr`.
- `strtok`
Trennt eine Zeichenkette in Teile.
- `strtolower`
Wandelt in Kleinbuchstaben um.
- `strtoupper`
Wandelt in Großbuchstaben um.

- `str_replace`
Ersetzt Zeichen einer Zeichenkette.
- `strtr`
Ersetzt mehrere Zeichen einer Zeichenkette durch mehrere andere.
- `substr`
Gibt einen Teil einer Zeichenkette zurück.
- `substr_count`
Ermittelt die Anzahl von Vorkommen in einer Zeichenkette.
- `substr_replace`
Ersetzt Teile einer Zeichenkette.
- `str_repeat`
Wiederholt ein bestimmtes Zeichen.
- `trim`
Entfernt Leerzeichen am Anfang und am Ende.
- `ucfirst`
Macht das erste Zeichen zum Großbuchstaben.
- `ucwords`
Das erste Zeichen jedes Wortes wird Großbuchstabe.
- `wordwrap`
Bricht Texte an Wortgrenzen um.

Einige Funktionen dienen auch der Verarbeitung natürlicher Sprache:

- `levenshtein`
Gibt die Distanz zweier Zeichenketten nach der Levenshtein-Funktion zurück. Eine Darstellung der Arbeitsweise finden Sie unter <http://odur.let.rug.nl/~kleiweg/lev/>.
- `metaphone`
Gibt den Lautwert einer Zeichenkette zurück. Informationen dazu sind unter <http://www.lanw.com/java/phonetic> zu finden.
- `soundex`
Bildet den Lautwert einer Zeichenkette ab.

Sprachfunktionen

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

- `similar_text`

Berechnet eine Ähnlichkeit zwischen Zeichenketten.

Zeichenkettenmanipulation

Leerzeichen entfernen

Bei der Übernahme von Daten spielen Leerzeichen eine wichtige Rolle. Viele Datenbanken haben feste Spaltenbreiten eingestellt und füllen Felder entsprechend mit Leerzeichen auf. Der Browser wird mehrere Leerzeichen nicht darstellen, außerdem stören führende Leerzeichen beim Sortieren. Es gibt also gute Gründe, überflüssige Leerzeichen zu entfernen. Dazu zählen alle so genannten Whitespaces⁷, neben dem Leerzeichen also beispielsweise auch der Tabulator.

`chop()` `ltrim()`, `trim()`

Die Funktion `chop` entfernt Leerzeichen vom Ende, `ltrim` entfernt vom Anfang der Zeichenkette an und `trim` an beiden Enden:

Listing 3.31:
string_choptrim:
Leerzeichen entfernen

```
<?php
$leerzeichen = " Am Anfang und am Ende ";
echo ".", chop($leerzeichen), "<BR>";
echo ".", ltrim($leerzeichen), "<BR>";
echo ".", trim($leerzeichen), "<BR>";
?>
```

Zeichenketten können vielfältig manipuliert werden. Beachten Sie dabei, dass es sich hier um Funktionen handelt und die geänderten Zeichenketten zurückgegeben werden. Das Argument selbst wird nicht verändert.

`strlen()` `strrev()` `strtolower()` `strtoupper()` `ucfirst()` `ucwords()`

Bei Zeichenketten können Sie die Länge mit `strlen` bestimmen. Die Reihenfolge der Zeichen lässt sich mit `strrev` umkehren. Eine Umwandlung zwischen Groß- und Kleinbuchstaben führen die Funktionen `strtolower`, `strtoupper`, `ucfirst` und `ucwords` aus. Das folgende Beispiel zeigt die Anwendung auf ein und dieselbe Zeichenkette:

Listing 3.32:
string_various:
Zeichenkettenmanipulationen

```
<?php
$zeichenkette = "Die Skriptsprache PHP ist vielseitig";
echo "strlen: ", strlen($zeichenkette) , " Zeichen.<BR>";
echo "strrev: ", strrev($zeichenkette) , "<BR>";
echo "strtolower: ", strtolower($zeichenkette) , "<BR>";
echo "strtoupper: ", strtoupper($zeichenkette) , "<BR>";
echo "ucfirst: ", ucfirst($zeichenkette) , "<BR>";
echo "ucwords: ", ucwords($zeichenkette) , "<BR>";
?>
```

Das Ergebnis sehen Sie in Abbildung 3.2. Experimentieren Sie mit diesen Befehlen, um ein Gefühl für die Verhaltensweise zu bekommen.

⁷ Als Whitespaces werden allgemein alle Zeichen bezeichnet, die beim Ausdruck »weiß« sind, also Leerzeichen, Tabulatoren, Zeilenumbrüche usw.

```
strlen: 36 Zeichen.
strrev: gitieSleiv tSi PHP ehcarpstpirkS eiD.
strtolower: die skriptSprache php ist vielseitig.
strtoupper: DIE SKRIPTSPRACHE PHP IST VIELSEITIG.
ucfirst: Die Skriptsprache PHP ist vielseitig.
ucwords: Die Skriptsprache PHP Ist Vielseitig.
```

Abbildung 3.2:
Ergebnis der
Zeichenketten-
manipulationen aus
Listing 3.32

Bei der Arbeit mit Zeichenketten werden diese häufiger zerlegt oder zusammengesetzt. Sie können dies besonders einfach mit den Funktionen `explode` und `implode` tun. `explode` zerlegt eine Zeichenkette anhand des Auftretens eines Trennzeichens und weist die einzelnen Elemente dann einem Array zu. Das folgende Beispiel zerlegt eine kommaseparierte Zeichenkette, sortiert das Array anschließend und gibt es wieder an die Zeichenkette zurück.

`explode()`
`implode()`

```
<?php
$zeichenkette = "Haide,Jörg,Uwe,Yvonne,Clemens,Lukas";
echo $zeichenkette, "<BR>";
$sorter = explode(",", $zeichenkette);
sort($sorter);
$zeichenkette = implode(",", $sorter);
echo $zeichenkette, "<BR>";
?>
```

Listing 3.33:
string_explode:
`explode` und
`implode`.

Der erste Parameter bestimmt das Trennzeichen, der zweite Parameter die umzuwandelnde Zeichenkette. Das Original wird nicht verändert.

Oft müssen Zeichenketten auch durchsucht werden. Auch hier kann wieder auf eine ganze Palette von Funktionen zugegriffen werden. Das folgende Beispiel zeigt wieder mehrere Anwendungsmöglichkeiten.

**Suche in
Zeichenketten**

Zeichenketten in PHP sind Null-basiert, das erste Zeichen hat also den Index 0. Sie werden im Gegensatz zu C nicht mit einem speziellen Zeichen abgeschlossen.



Mit `strpos` und `strrpos` wird die Position eines Zeichens in einer Zeichenkette bestimmt. Wenn Sie gleich an dieser Stelle die Zeichenkette trennen möchten, bieten sich `strchr`, `substr`, `strstr` und `ereg` an.

`strpos()`
`strrpos()`

`strstr` sucht das Auftreten eines Zeichens vom Beginn an und gibt die Zeichenkette ab dieser Position zurück. `strrchr` sucht das letzte Auftreten des Zeichens und gibt die Zeichenkette von dort beginnend bis zum Ende zurück. `substr` ermöglicht eine Auswahl eines Teils einer Zeichenkette durch Angabe der Position und der Länge des Teils.

`strchr()`
`substr()`
`strstr()`

```
<?php
$zeichenkette = "Haide,Jörg,Uwe,Yvonne,Clemens,Lukas";
echo "$zeichenkette <P>";
echo "Erste Pos.: ", strpos($zeichenkette,"e"), "<br>";
echo "Letzte Pos.: ", strrpos($zeichenkette,"e"), "<br>";
echo "Teil: ", substr($zeichenkette, 6, 4), "<br>";
echo "Von links: ", strchr($zeichenkette, "r"), "<br>";
```

Listing 3.34:
string_strpos:
Zeichenketten-
manipulationen

```
echo "Von rechts: ",strstr($zeichenkette, "r"), "<br>";
?>
```

Die Ausgabe finden Sie in der folgenden Abbildung. Manipulieren Sie die Werte, um die Reaktion der Funktionen besser zu verstehen.

Abbildung 3.3:
Zeichenketten-
manipulationen aus
Listing 3.34

```
Haide,Jörg,Uwe,Yvonne,Clemens,Lukas
Erste Pos.: 4
Letzte Pos.: 26
Teil: Jörg
Von links: rg,Uwe,Yvonne,Clemens,Lukas
Von rechts: rg,Uwe,Yvonne,Clemens,Lukas
```

Versuchen Sie *genau* nachzuvollziehen, wie die Funktionen im letzten Beispiel arbeiten. Zeichenkettenfunktionen werden sehr häufig benötigt und die genaue Kenntnis erleichtert später die Programmierung komplexer Projekte.

strspn() strcspn()

strcspn und strspn suchen nach Zeichen und geben den Teil der Zeichenkette zurück, für eine Übereinstimmung gefunden bzw. nicht gefunden wurde.

Listing 3.35:
string_strspn:
Zeichenketten
untersuchen

```
<?php
$zeichenkette = "Haide,Jörg,Uwe,Yvonne,Clemens,Lukas";
echo "$zeichenkette <P>";
echo "Enthalten: ", strspn($zeichenkette,"ediaH"), "<br>";
echo "Nicht enth.: ", strcspn($zeichenkette,"ö");
?>
```

Im Beispiel gibt die erste Funktion 5 zurück – die ersten fünf Zeichen sind in der Musterzeichenkette enthalten. Die zweite Funktion dagegen gibt 7 zurück; hier wird die Anzahl der nicht übereinstimmenden Zeichen ermittelt. Die erste Übereinstimmung »ö« ist auf Position 8, deshalb stimmen die ersten sieben Zeichen nicht überein. Die Reihenfolge der Zeichen in der Suchzeichenfolge spielt keine Rolle, »Haide« und »ediaH« sind also äquivalent verwendbar.

Ersetzen von Zeichenketten str_replace()

Neben dem Suchen ist auch das Ersetzen von Teilen einer Zeichenkette in fast jedem Skript zu finden. Sie können dafür die Funktion str_replace verwenden. Das folgende Beispiel gibt eine durch Kommata getrennte Liste zeilenweise aus, indem als Trennzeichen das HTML-Tag
 eingesetzt wird:

Listing 3.36:
string_strreplace
Teile einer Zeichen-
kette ersetzen

```
<?php
$zeichenkette = "Haide,Jörg,Uwe,Yvonne,Clemens,Lukas";
echo "$zeichenkette <p>";
echo str_replace(",","<br>",$zeichenkette);
?>
```

Die Abbildung zeigt die Wirkung bei der Ausgabe der Zeichenkette.

```
Haide,Jörg,Uwe,Yvonne,Clemens,Lukas

Haide
Jörg
Uwe
Yvonne
Clemens
Lukas
```

Abbildung 3.4:
Ersatz zur HTML-
Formatierung:
Listing 3.36

Umwandlungs- und Verschlüsselungsfunktionen

Typische Umwandlungsfunktionen sind `chr` und `ord`. Damit wandeln Sie einzelne Zeichen in ASCII-Werte um und umgekehrt. Dabei wird der erweiterte 8-Bit-ASCII-Zeichensatz berücksichtigt, sodass auch Umlaute korrekt behandelt werden, wie im folgenden Beispiel zu sehen ist:

`chr()`
`ord()`

```
<?php
$zeichen = "ö";
echo "ASCII-Wert von $zeichen = ", ord($zeichen);
echo "<br>";
echo "Zeichen 105 = ", chr(105);
?>
```

Listing 3.37:
`string_ordchr`:
ASCII-Werte

Oft wird `chr` verwendet, um bestimmte Zeichen auszugeben, die vom PHP-Parser von Bedeutung sind, beispielsweise Anführungszeichen. Schauen Sie sich in diesem Zusammenhang auch die Funktion `sprintf` in ➔ Abschnitt 3.2.9 *Formatierfunktionen* ab Seite 201 an.

Sonderzeichenbehandlung

In vielen Fällen werden Sie auf Zeichen stoßen, die innerhalb einer Zeichenkette eine besondere Bedeutung haben. Fast in jedem Skript wurde bereits diese Eigenschaft genutzt. Offensichtlich ist der PHP-Interpreter in der Lage, Variablen anhand des `$`-Zeichens mitten in einer Zeichenkette zu erkennen und durch den Inhalt zu ersetzen. Was passiert aber, wenn Sie tatsächlich ein `$`-Zeichen ausgeben möchten?

Die Funktionen `addslashes`, `quotemeta` und `stripslashes` helfen hier weiter. Die Funktion `quotemeta` setzt einen Backslash (`>\«`-Zeichen) vor jedes der folgenden Sonderzeichen:

`quotemeta()`
`addslashes()`
`stripslashes()`

```
. \ + * ? [ ^ ] ( $ )
```

Mit `addslashes` werden dagegen Sonderzeichen behandelt, die oft von Datenbanken interpretiert werden:

```
' " \ NUL
```

Dabei steht `NUL` für den absoluten Wert 0. In einigen Programmiersprachen dient ein NUL-Byte als Endekennzeichen für Zeichenketten. Die Funktion `stripslashes` macht die durch `addslashes` vorgenomme-

nen Umwandlungen wieder rückgängig. Der Vorgang wird übrigens auch oft als »escapen« bezeichnet.

Weitere Umwandlungen sind HTML-spezifisch. Wegen der herausragenden Bedeutung bei der Webserverprogrammierung ist ihnen der ➔Abschnitt *HTML-spezifische Funktionen* ab Seite 199 gewidmet.

Zeichenketten vergleichen

strcmp()

Vergleiche zwischen Zeichenketten sind oft nur schwer praktisch umzusetzen. `strcmp` vergleicht zwei Zeichenketten und gibt einen Wert -1 zurück, wenn die erste Zeichenkette kleiner als die zweite ist, oder +1 im umgekehrten Fall oder 0, wenn beide Zeichenketten gleich sind. Als »kleiner« wird die Zeichenkette angesehen, wenn der ASCII-Werte des ersten nicht übereinstimmenden Zeichens auf einer Position in der Zeichenkette kleiner ist.

Da Kleinbuchstaben im ASCII-Alphabet nach den Großbuchstaben folgen, ist »C« kleiner als »a«. Der direkte Vergleich ist natürlich wenig praxisnah. PHP kennt deshalb Funktionen, die auf Ähnlichkeit prüfen oder sogar Lautwerte ermitteln.

Listing 3.38:
string_strcmp:
Anwendung der
Funktion strcmp

```
<?php
$str1 = "Clemens";
$str2 = "Martin";
echo strcmp($str1, $str2);
?>
```

strcasecmp()

Die Funktion `strcasecmp` arbeitet ebenso, berücksichtigt aber Groß- und Kleinschreibung nicht. Sie ist erst ab PHP 4 verfügbar.

soundex()

Das folgende Skript zeigt die Anwendung der Funktion `soundex`, die ein Lautäquivalent berechnet:

Listing 3.39:
string_soundex:
Anwendung von
soundex

```
<?php
$str1 = "Meier";
$str2 = "Mayer";
$str3 = "Maher";
echo "Soundex-Wert 1: ", soundex($str1);
echo "<br>";
echo "Soundex-Wert 2: ", soundex($str2);
echo "<br>";
echo "Soundex-Wert 3: ", soundex($str3);
?>
```

Die drei in Listing 3.39 gezeigten Zeichenketten geben alle den Wert M600 zurück. Bei einer Volltextsuche würden diese als »ähnlich klingende« Schreibweisen einander entsprechen und zur Erfüllung der Suchbedingung führen.

HTML-spezifische Funktionen

HTML selbst ist eine Seitenbeschreibungssprache, die ASCII-Zeichen verwendet. Dokumente werden mit Elementen strukturiert, die als Tags⁸ bezeichnet werden. Solche Tags beginnen mit einem <-Zeichen und enden mit einem >-Zeichen. Einige Symbole, wie beispielsweise das &-Zeichen, haben ebenfalls eine besondere Bedeutung. Mit speziellen Funktionen wird die Umwandlung unterstützt.

Die Funktion `htmlspecialchars` wandelt die folgenden Zeichen so um, dass die Darstellung in HTML korrekt erfolgt: **htmlspecialchars()**

- & wird als `&` dargestellt.
- " wird als `"` zurückgegeben.
- < wird in `<` umgewandelt.
- > wird als `>` ausgegeben.

Diese Funktion berücksichtigt keineswegs alle HTML-Zeichen, sondern tatsächlich nur die hier aufgeführten. Die Anwendung betrifft vor allem Formulare, wo diese Zeichen gesondert verarbeitet werden können.



Eine mögliche Anwendung sehen Sie im folgenden Listing, das den Code und seine Ausführung in HTML wiedergibt:

```
<?php
$html = "Wir haben mit <br><b>PHP</b><br> was vor.";
echo $html;
echo "<p>";
echo htmlspecialchars($html);
?>
```

Listing 3.40:
html_specialchars:
HTML-Tags
umwandeln

Abbildung 3.5 zeigt die Auswirkung bei der Ausgabe auf einem Browser.

```
Wir haben mit
PHP
was vor.
Wir haben mit <br><b>PHP</b><br> was vor.
```

Abbildung 3.5:
Umwandlung von
HTML-Zeichen

Wenn Sie alle HTML-Symbole umwandeln möchten, die es gibt, dann verwenden Sie besser die Funktion `htmlentities`. Diese berücksichtigt auch die Umlaute. Dabei wird beispielsweise das Ä in `Ä` verwandelt. Um Sonderzeichen in Zeichenketten zu erkennen, wird der ISO-8859-1-Zeichensatz verwendet.

htmlentities()

⁸ Vom englischen Wort »Tag«, was etwa Marke oder Zeichen bedeutet.

Die Umwandlungstabelle für die HTML-Sonderzeichen kann mit `get_html_translation_table` ausgelesen werden. Ein Anwendungsbeispiel finden Sie in ➔Abschnitt 3.6 *Hilfsfunktionen* ab Seite 294.

n12br()

Zeichenketten enthalten oft normale Zeilenumbrüche, die Darstellung ist teilweise vom Betriebssystem abhängig. In HTML werden Zeilenumbrüche mit dem Tag `
` erzeugt. Die Funktion `n12br` (newline to break) fügt an alle Zeilenumbrüche des Parameters die Zeichenfolge »`
`« an:

```
echo n12br($vieltext);
```

Damit die Ausgabe auch bei der Anzeige des Quelltextes einer HTML-Seite nicht unleserlich wird, bleiben die ursprünglichen Umbruchcodes (`chr(10)` und/oder `chr(13)`) erhalten. Auf welche Art Zeilenumbrüche `n12br` reagiert, hängt vom Betriebssystem ab. Wenn Sie Ihre Texte unter Linux verwalten und `n12br` auf NT ablaufen lassen, wird die Funktion nicht unbedingt richtig reagieren.



Aus der HTML-Programmierung kennen Sie sicher `
`. Die Ausgabe von `
` entspricht dem Standard XHTML 1.0. Damit ist PHP für die Zukunft gerüstet, falls Browser einmal auf die Einhaltung dieses Standard bestehen. Ältere Browser stören sich an dem schließenden Schrägstrich übrigens nicht. Die gilt im übrigen auch für andere von PHP intern erzeugten Tags.

Codierung von URL-Daten

rawurldecode() rawurlencode() urldecode() urlencode()

Eine andere Situation tritt auf, wenn Daten per URL übertragen werden. Auf diese Methoden wird in Kapitel 4 ausführlich eingegangen. Wenn Sie Zeichenketten an die URL anhängen, müssen diese den Konventionen für den Aufbau einer URL genügen. Umlaute, Sonderzeichen und Leerzeichen sind dabei nicht erlaubt. Mit Hilfe der Funktion `rawurlencode` werden die entsprechenden zulässigen Zeichen verwandelt. Dabei handelt es sich um die Zeichenfolge `%HH`, wobei `HH` für den hexadezimalen ASCII-Wert des speziellen Zeichens steht. Die Anwendung können Sie dem folgenden Listing entnehmen:

Listing 3.41:
html_urldecode: Die
Funktionen
rawurlencode und
rawurldecode

```
<?php
$html = "Dies soll übertragen werden";
$html1 = rawurlencode($html);
$html2 = urlencode($html);
echo $html;
echo "<p>";
echo $html1;
echo "<br>";
echo $html2;
?>
```

Das Ergebnis sieht etwa folgendermaßen aus:

```
Dies soll übertragen werden
Dies%20soll%20%Fbertragen%20werden
Dies+soll+%Fbertragen+werden
```

Abbildung 3.6:
Ausgabe des Skripts
aus Listing 3.41

Die technische Grundlage für den Umwandlungsvorgang können Sie in der RFC 1738 nachlesen. Umgewandelt werden sollen danach alle Zeichen außer alphanumerischen Zeichen und dem Punkt ».«, dem Unterstrich »_« und dem Minuszeichen »-«. In der Behandlung des Leerzeichens unterscheiden sich die beiden Funktionen `rawurlencode` und `urlencode` übrigens. `urlencode` nimmt hier das Pluszeichen als Ersatzzeichen, was historisch verwendet wird und aus Kompatibilitätsgründen zum Einsatz gelangt. Es entspricht aber nicht der bereits erwähnten RFC 1738.

RFC 1738

Der Ordnung halber sei an dieser Stelle auch die Funktion `parse_str` erwähnt, die an die URL angehängte Parameter erkennt und Variablen zuweist. Diese Funktion ist für alle Skripte, die interaktiv Daten verarbeiten, von größter Bedeutung. Sie finden eine ausführliche Beschreibung in ➡ Kapitel 4 *Interaktive Webseiten* ab Seite 299.

`parse_str()`

3.2.9 Formatierfunktionen

Daten müssen häufig vor der Ausgabe auf dem Browser formatiert werden. Dazu stehen die Funktionen `printf` und `sprintf` zur Verfügung. Beide funktionieren äquivalent. `sprintf` gibt das Ergebnis an eine Zeichenkette zurück, `printf` sendet das Ergebnis dagegen direkt an den Zeichenausgabepuffer. Wenn PHP im Webserver arbeitet, erfolgt die Ausgabe via CGI-Schnittstelle auf dem Browser.

Universelle Zeichenkettenformatierungen

Beide Funktionen sind in der Anwendung identisch, die folgende Beschreibung bezieht sich deshalb nur auf `sprintf`. Die Funktion hat mindestens zwei Parameter. Der erste gibt eine Formatieranweisung an, der zweite (und alle folgenden) gibt einen Wert an, der in die Formatieranweisung eingesetzt wird. Hier ein Beispiel:

**`printf()`
`sprintf()`**

```
$result = sprintf("%04d-%02d-%02d", $year, $month, $day);
```

Eine komplette Formatieranweisung besteht aus fünf Elementen:

1. Ein Füllzeichen

Der Standardwert ist das Leerzeichen. Wenn Sie mehrere Stellen wünschen und der Zahlen- oder Zeichenkettenwert diese nicht erreicht, wird der Rest mit dem Füllzeichen aufgefüllt.



Beispiel

2. *Ausrichtung*

Standardmäßig werden alle Werte rechts ausgerichtet, die Füllzeichen also als führende Zeichen behandelt. Mit einem Minuszeichen wird diese Funktion umgedreht.

3. *Zeichenzahl*

Eine optionale Anzahl Zeichen, die für den Wert in der Ausgabe reserviert wird. Fehlende Zeichen werden mit Füllzeichen aufgefüllt.

4. *Dezimale*

Für Gleitkommazahlen kann die Anzahl der Dezimalstellen bestimmt werden. Auf andere Werte hat dies keinen Einfluss.

5. *Typ*

Der Typ gibt an, wie der Wert generell behandelt wird. Folgende Typen sind möglich:

- %

Gibt ein Prozentzeichen aus.

- b

Ganze Zahl (Integer), Darstellung in binärer Form.

- c

Ganze Zahl (Integer), Darstellung als Zeichen aus dem ASCII-Zeichensatz.

- d

Ganze Zahl (Integer), Darstellung als Dezimalzahl.

- f

Gleitkommazahl, Darstellung in Exponentialform.

- o

Ganze Zahl (Integer), Darstellung in oktaler Form.

- s

Zeichenkette.

- x

Ganze Zahl (Integer), Darstellung in hexadezimaler Form. Die hexadezimalen Ziffern werden als Kleinbuchstaben »a« bis »f« dargestellt.

- X

Ganze Zahl (Integer), Darstellung in hexadezimaler Form. Die hexadezimalen Ziffern werden als Großbuchstaben »A« bis »F« dargestellt.

Das folgende Skript zeigt verschiedene Anwendungsbeispiele:

```
<?php
$money = 63.14;
printf("DM %0.2f <br>", $money);
$integer = 34;
printf("[%04d] <br>", $integer);
$percent = 14.45;
printf("Der Anteil beträgt %0.2f%%", $percent);
?>
```

Listing 3.42:
printf: Verschiedene
Formen der printf-
Funktion

Sie können in der Formatierzeichenkette beliebige Kombinationen aus solchen Zeichen darstellen, beispielsweise für die Ausgabe von Währungen. Alle Zeichen, die nicht zur Formatieranweisung gehören, werden unverändert ausgegeben, wie im letzten Beispiel das Präfix »DM«. Für die Ausgabe von führenden Nullen ist die Funktion ebenfalls geeignet.

```
DM 63.14
[0034]
Der Anteil beträgt 14.45%
```

Abbildung 3.7:
Ausgabe des Skripts
aus Listing 3.42

Dieselben Formatieranweisungen verwendet auch die Funktion `sscanf`, um Zeichenfolge auf bestimmte Muster hin zu durchsuchen. Werden diese erkannt, überträgt die Funktion den entsprechenden Teil in eine Variable. Das folgende Beispiel zeigt die Aufbereitung eines amerikanischen Datumsformates und die Ausgabe in der in Deutschland üblichen Form mit `sscanf` und `printf`.

sscanf()

```
<?php
$date = '2001/26/5';
$d = sscanf($date, '%d/%d/%d');
printf('%02d.%02d.%4d', $d[1], $d[2], $d[0]);
?>
```

Listing 3.43:
sscanf: Daten mit
sscanf scannen

Das von `sscanf` erzeugte Array `%d` hat so viele Elemente, wie gültige Formatierungen entdeckt werden konnten. Dies muss nicht der Anzahl der Formatanweisungen entsprechen.

```
26.05.2001
```

Abbildung 3.8:
Ausgabe des Skripts
aus Listing 3.43

number_format()

Geht es bei der Ausgabe jedoch nur um Zahlen, werden Sie vielleicht mehr mit `number_format` anfangen können. Diese Funktion eignet sich vor allem, um das intern verwendete englische Zahlenformat in das in Deutschland übliche umzuwandeln.

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Listing 3.44:
numberformat:
 Zahlen formatiert
 ausgeben

```
<?php
$number = 34009.5677;
echo number_format($number, 2, ',', '.');
?>
```

Auch diese Funktion ist in der Lage, korrekt zu runden, wenn es die Anzahl Dezimalstellen erfordert. Übergeben werden neben der Zahl (erster Parameter) die Anzahl der Dezimalstellen (im Beispiel 2), das als Komma verwendete Zeichen und optional ein Zeichen, das als Tausendertrenner dient.

Abbildung 3.9:
 Zahlenformatierung

**Mehr in der
gedruckten
Referenz**

34.009,57

PHP verfügt über sehr viele Zeichenkettenfunktionen. Sie sollten sich neben den hier exemplarisch vorgestellten auch im Referenzbuch informieren, wo weitere Beispiele zu finden sind. Die gedruckte Referenz zu PHP 4 erschien unter dem Titel »PHP 4. Die Referenz« bei Carl Hanser mit der ISBN-Nummer 3-446-21687-1.

3.2.10 Mathematische Funktionen

PHP verfügt über alle in höheren Programmiersprachen üblichen mathematischen Funktionen. Die Anwendung birgt kaum Schwierigkeiten, deshalb soll hier die Darstellung in Tabellenform genügen.

Tabelle 3.7:
 Mathematische
 Funktionen

abs()
acos()
asin()
atan()
atan2()
cos()
exp()
max()
min()
log()
log10()
pow()
sin()
sqrt()
tan()

Funktion	Beschreibung
abs(\$x)	Absoluter Betrag
acos(\$x)	Arcus Cosinus
asin(\$x)	Arcus Sinus
atan(\$x)	Arcus Tangens
atan2(\$x)	Arcus Tanges Hyperbolicus
cos(\$x)	Cosinus
exp(\$x)	e ^x , Potenz zur Basis e (Eulersche Zahl)
max(\$x[args])	Maximalwert der Argumenteliste
min(\$x[args])	Minimalwert der Argumenteliste
log(\$x)	Natürlicher Logarithmus
log10(\$x)	Dekadischer Logarithmus
pow(\$x, \$y)	Potenzfunktion, x ^y
sin(\$x)	Sinus
sqrt(\$x)	Quadratwurzel

Funktion	Beschreibung
<code>tan(\$x)</code>	Tangens

Die Umwandlungsfunktionen dienen der Bearbeitung von Zahlenwerten. So können Sie die Zahlenbasis wechseln und Zahlen nach verschiedenen Verfahren runden.

Funktion	Beschreibung
<code>floor(fliesskommazahl)</code>	Ganzzahliger Teil einer Zahl
<code>ceil(fliesskommazahl)</code>	Nächsthöhere Ganzzahl
<code>round(fliesskommazahl[,stelle])</code>	Rundung auf Stellenzahl <i>stelle</i>
<code>base_convert (nummer, ausgangsbasis, zielbasis)</code>	Wandelt von einem beliebigen Zahlensystem der Basis 2 bis 36 in ein anderes um
<code>bindec(binaerwert)</code>	Binär -> Dezimal
<code>decbin(dezimalwert)</code>	Dezimal -> Binär
<code>dechex(dezimalwert)</code>	Dezimal -> Hexadezimal
<code>decoct(dezimalwert)</code>	Dezimal -> Oktal
<code>hexdec(hexwert)</code>	Hexadezimal -> Dezimal
<code>octdec(oktalwert)</code>	Oktal -> Dezimal

Tabelle 3.8:
Umwandlungs-
funktionen

floor(), ceil()
base_convert()
bindec()
decbin()
dechex()
decoct()
hexdec()
octdec()
round()

Wenn Sie Logarithmen mit einer beliebigen Basis berechnen müssen, verwenden Sie folgende Definition. Das erste Argument ist die zu berechnende Mantisse, das zweite die Basis des Logarithmus:

```
<?php
function logx($mant, $raise) {
    return log($mant)/log($raise);
}
?>
```

Listing 3.45:
math_logx:
Logarithmus
berechnen

Eine Besonderheit ist bei `round` zu beachten. Die Rundung erfolgt nicht in allen Fällen korrekt. Das liegt an der internen Darstellung der Zahlen. Einige einfache Werte, wie beispielsweise 11,5, sind nicht exakt im Binärformat darstellbar. Statt dessen speichert der Computer 11,499999999999. Die Rundungsfunktion rundet dann auf 11 ab.

Das Beispiel *math_functions.php* zeigt die Reaktion der Funktionen `ceil`, `floor` und `round` auf bestimmte Eingabewerte. Der Ausgabe können Sie dies entnehmen:

**Probleme mit
round**

Abbildung 3.10:
Mathematische
Funktion in Aktion

Funktion	1.22	1.99	-17	-11.5
floor()	1	1	-17	-12
ceil()	2	2	-17	-11
round()	1	2	-17	-12
Umwandlungsfunktionen				
	DecBin()	DecHex()	DecOct()	Dezimal
Dec->	1111010	7a	172	122
Zahlenbasen				
	Basis 2	Basis 6	Basis 12	Basis 16
base_convert-> 45	101101	113	39	2d

Mathematische Konstanten

In PHP 3 stand nur die Konstante π (PI) zur Verfügung. Ab PHP 4 gibt es eine ganze Palette häufiger benötigter Konstanten, die in der folgenden Tabelle zusammengefasst sind.

Tabelle 3.9:
Mathematische
Konstanten und
deren Definition

Konstante	Exakter Wert	Definition
M_PI	3.14159265358979323846	Der Wert π (Pi)
M_E	2.7182818284590452354	e (Eulersche Zahl)
M_LOG2E	1.4426950408889634074	$\log_2 e$
M_LOG10E	0.43429448190325182765	$\log_{10} e$
M_LN2	0.69314718055994530942	$\log_e 2$
M_LN10	2.30258509299404568402	$\log_e 10$
M_PI_2	1.57079632679489661923	$\pi/2$
M_PI_4	0.78539816339744830962	$\pi/4$
M_1_PI	0.31830988618379067154	$1/\pi$
M_2_PI	0.63661977236758134308	$2/\pi$
M_2_SQRTPI	1.12837916709551257390	$\frac{2}{\sqrt{\pi}}$
M_SQRT2	1.41421356237309504880	$\sqrt{2}$
M_SQRT1_2	0.70710678118654752440	$\frac{1}{\sqrt{2}}$

Zufallszahlen

Zufallszahlen werden häufig benötigt, um Vorgänge zu steuern oder beispielsweise Kennwörter zu erzeugen. Zufallsfolgen beruhen auf mathematischen Funktionen, die zwar einen chaotischen Verlauf haben, aber dennoch einer strengen Folge gehorchen, sie sind pseudozufällig. Die Zufälligkeit wird erst erzeugt, wenn der Startwert variiert.

Die folgende Tabelle zeigt Funktionen zum Abruf der Zufallswerte und zum Setzen des Startwertes.

Funktion	Beschreibung
<code>srand(\$x)</code>	Setzt den Startwert für den Zufallsgenerator.
<code>rand([\$min][,\$max])</code>	Gibt eine Zufallszahl zwischen 0 und 1 oder, wenn benutzt, zwischen <i>\$min</i> und <i>\$max</i> zurück.
<code>getrandmax()</code>	Gibt die höchstmögliche Zahl an, die <code>rand</code> zurückgeben kann.
<code>mt_srand</code>	Setzt den Startwert für den Zufallsgenerator.
<code>mt_rand([\$min], [\$max])</code>	Gibt eine Zufallszahl zwischen 0 und 1 oder, wenn angegeben, zwischen <i>\$min</i> und <i>\$max</i> zurück.
<code>mt_getrandmax</code>	Gibt die höchstmögliche Zahl an, die <code>mt_rand</code> zurückgeben kann.

Tabelle 3.10:
Funktionen für
Zufallszahlen

srand()
rand()
mt_srand()
mt_rand()
mt_getrandmax()
getrandmax()

Die Funktionen mit dem Präfix `mt_` sind neuer und sollten grundsätzlich bevorzugt werden. Die in den ursprünglichen PHP-Bibliotheken verwendeten Zufallsgeneratoren sind relativ unbekannt und langsam. Vor allem beim Einsatz in der Kryptografie ist die Charakteristik eines Zufallsgenerators wichtig. Die `mt`-Funktionen wurden von Mersenne Twister (`mt`) entwickelt und sind gut dokumentiert. Mehr Informationen dazu finden Sie auf den Webseiten:

mt-Funktionen

<http://www.math.keio.ac.jp/~matumoto/emt.html>

<http://www.scp.syr.edu/~marc/hawk/twister.html>

3.2.11 Zeit- und Datumsfunktionen

Datumsberechnungen nehmen in der praktischen Programmierung breiten Raum ein. Entsprechend umfangreich ist die Unterstützung in PHP.

Kalenderfunktionen

Auf eine umfassende Darstellung der Kalenderfunktionen soll hier verzichtet werden. Die meisten Funktionen dürften bestenfalls Historiker interessieren. Die Liste sollten Sie aber im Hinterkopf behalten, falls Sie irgendwann auf ein entsprechendes Problem stoßen. Denn an Datumsberechnungen kann man sich wirklich die Zähne ausbeißen. Eine gute Quelle für weitere Informationen ist die folgende Adresse:

<http://genealogy.org/~scottlee/cal-overview.html>

Tabelle 3.11:
Kalenderfunktionen

Funktion	Beschreibung
jdtogregorian	Konvertiert ein Datum des Julianischen Kalenders in das gregorianische Format
gregoriantojd	Konvertiert das gregorianische Format in ein Datum des Julianischen Kalenders
jdtojulian	Wandelt ein julianisches Datum in einen julianischen Tageswert um (für Berechnungen mit diesem)
juliantojd	Wandelt einen julianischen Tageswert wieder in die entsprechende Datumsangabe zurück
jdtojewish	Wandelt einen julianischen Tageswert in ein Datum des jüdischen Kalenders um
jewishtojd	Die entsprechende Umkehrfunktion zu jdtojewish
jdtofrrench	Wandelt ein julianisches Datum in ein Datum des Kalenders der französischen Republik um
frrenchtojd	Die Umkehrfunktion zu jdtofrrench
jdmonthname	Diese Funktion gibt einen Monatsnamen zurück, die Datumsangabe muss dem Julianischen Kalender entsprechen. Die Ausgabe kann jedoch durch einen zweiten Parameter gesteuert werden, der den Ursprungskalender bestimmt.
jddayofweek	Gibt den Wochentag zurück, entweder als Zahl oder als Wort (aus dem englisch-gregorianischen Kalender)
easter_date	Die Funktion gibt den UNIX-Zeitcode für Ostersonntag des angegebenen Jahres zurück.
easter_days	Gibt die Anzahl der Tage aus, die Ostern nach dem 21. März des angegebenen Jahres liegt.

easter_date()
easter_days()

Die Oster-Funktionen sind zumindest eine kurze Betrachtung wert, da derartige Kalenderberechnungen nicht trivial sind und vom Ostertermin viele andere Feiertage abhängen. Zuerst müssen Sie sich die Definition der UNIX-Zeitcodes in Erinnerung rufen. Die Unix-Welt begann 1970 und die Definition lässt Daten bis 2037 zu. Wollen Sie Daten vor oder nach diesen Jahren berechnen, können Sie `easter_date` nicht einsetzen. Die Funktion `easter_days` erschwert zwar unter Umständen die Anwendung, ist aber unabhängig von derartigen Grenzdaten. Leicht zu beantworten ist die Frage, auf welches Datum Ostern im Jahr 2002 fällt (vielleicht wissen Sie es, weil Sie dieses Buch im größten Osterei gefunden haben):

```
<?php
setlocale(LC_TIME, 'de');
echo 'Ostern ist am ' . strftime("%d.%m.%Y", easter_date(2002));
?>
```

Listing 3.46:
easter_date: Das
Osterdatum für das
Jahr 2002 ermitteln

Vielleicht mögen Sie sich fragen, welche Bedeutung der 21. März hat, der in der Funktion `easter_days` zur Anwendung kommt. Diesem Datum liegt die Definition des Osterfestes zugrunde. Ostern ist per Definition am ersten Sonntag nach dem ersten Vollmond nach dem Frühlingsanfang. Und Frühlingsanfang ist am 21. März. Wenn Vollmond auf diesen Tag fällt und es zugleich ein Sonntag ist, wäre am 21. März Ostern – der frühestmögliche Termin.

Wie man Ostern berechnet

Datumsfunktionen

Wichtiger als die Kalenderfunktionen sind allgemeine Datumsberechnungen. Tabelle 3.12 gibt eine Übersicht über die in PHP verfügbaren Datumsfunktionen.

Funktion	Beschreibung
<code>checkdate</code>	Gibt TRUE zurück, wenn das angegebene Datum korrekt ist. Benötigt drei Argumente für Monat, Tag und Jahr (in dieser Reihenfolge).
<code>date</code>	Formatiert ein Datum für die Ausgabe.
<code>getdate</code>	Gibt ein assoziatives Array mit Datums- und Zeitangaben zurück.
<code>gmdate</code>	Wie <code>date</code> , berücksichtigt aber GMT.

Tabelle 3.12:
Funktionen zum
Umgang mit
Datumswerten

Für die Arbeit mit Daten ist oft das aktuelle Datum von großer Bedeutung. Beachten Sie jedoch, dass der Server sich immer auf sein eigenes Systemdatum bezieht. Wenn Sie PHP in den USA hosten lassen, wird das Ergebnis nicht unbedingt den Erwartungen entsprechen, wenn Sie Ihre Nutzer zeitabhängig begrüßen möchten.

Für die Darstellung eines Datums gibt es viele Formate. Die Funktion `date` gibt ein Datum formatiert zurück, so dass Sie lokale Besonderheiten leicht berücksichtigen können. Die Funktion benötigt zwei Argumente, eine Formatieranweisung und eine Zeitinformation als Unix-Zeitstempel. Wenn der zweite Parameter entfällt, wird die aktuelle Zeit genommen:

date()

```
echo date("l dS of F Y");
```

Innerhalb der Formatieranweisung sind folgende Symbole von Bedeutung:

Formatsymbole

- a
»am« oder »pm«

- A
»AM« oder »PM«
- B
Swatch-Internet-Zeit
- d
Tag des Monats mit 2 Stellen und führender Null: »01« bis »31«
- D
Tag der Woche als Abkürzung mit drei Buchstaben: »Mon«
- F
Monat, ausgeschrieben: »September«
- h
Stunde im 12-Stunden-Format: »01« bis »12«
- H
Stunde im 24-Stunden-Format: »00« bis »23«
- g
Stunde im 12-Stunden-Format ohne führende Null: »1« bis »12«.
- G
Stunde im 24-Stunden-Format ohne führende Null: »0« bis »23«.
- i
Minuten: »00« bis »59«.
- I (großes »i«)
»1« bei Sommerzeit, »0« bei Winterzeit (Boolescher wert)
- j
Tag des Monats ohne führende Null: »1« bis »31«
- l (kleine »L«)
Wochentag, voll ausgeschrieben: »Friday«
- L
Boolescher Wert, der bestimmt, ob das Datum in einem Schaltjahr liegt: »0« oder »1«
- m
Monat mit führender Null: »01« bis »12«

- n
Monat ohne führende Null: »1« bis »12«
- M
Monat als Abkürzung: »Jan«
- s
Sekunden mit führender Null: »00« bis »59«
- S
Das Suffix der englischen Ordnungszahlen: »th«, »nd« usw.
- T
Einstellung der Zeitzone des Servers: »GMT« oder »MET« usw.
- t
Anzahl der Tage in einem Monat: »28« bis »31«
- U
Sekunden seit Beginn der Unix-Epoche (1.1.1970)
- w
Numerische Darstellung des Wochentags: »0« (für Sonntag) bis »6« (Samstag)
- Y
Vierstellige Ausgabe des Jahres: »1999« oder »2000«
- y
Zweistellige Ausgabe des Jahres: »99« oder »00«
- z
Tag im Jahr: »0« bis »365«
- Z
Differenz zur Zeitzone in Sekunden: »-43200« bis »43200«. Dies entspricht -12 oder +12 Stunden.

Alle anderen Zeichen werden ignoriert und unverändert ausgegeben.
Hier zwei Beispiele:

```
<?php
echo date( "l dS of F Y h:i:s A" );
echo '<br>';
echo "The 1st of July 2000 is a ";
echo date("l", mktime(0,0,0,7,1,2000));
?>
```

Listing 3.47:
Datums-
formatierungen:
dateform

Die Abbildung zeigt das Resultat, das noch nicht perfekt ist. Schauen Sie sich für Ausgaben von Namen in deutscher Sprache auch die Funktionen `setlocale` und `strftime` an. Mit `setlocale` setzen Sie ein Sprachformat. `strftime` arbeitet ähnlich wie `date`, nutzt aber andere Formatsymbole. Mehr zu `setlocale` finden auf ➡ Seite 215.

Das Beispiel in Listing 3.48 nutzt dies gleich.

Abbildung 3.11:
Ausgabe von Listing
3.47

```
Friday 03rd of August 2001 06:19:27 PM
The 1st of July 2000 is a Saturday
```

Datumsberechnungen

Manchmal müssen mit Daten Berechnungen angestellt werden. Die folgenden Beispiele zeigen, wie man das mit den gezeigten Funktionen leicht erledigen kann:

Listing 3.48:
Datumsberechnungen im Beispiel
datecalc

```
<?php
$morgen = mktime(0,0,0,date("m"), date("d")+1,date("Y"));
$letztermonat = mktime(0,0,0,date("m")-1, date("d"), date("Y"));
$naechstesjahr = mktime(0,0,0,date("m"), date("d", date("Y")+1));
setlocale(LC_TIME, "ge");
printf ("Morgen ist %s, der letzte Monat war ein %s.",
        strftime('%A', $morgen),
        strftime('%B', $letztermonat));
?>
```

`mktime` gibt den Zeitstempel für ein Datum zurück und wird im nächsten Abschnitt vorgestellt, ebenso werden `strftime` und `setlocale` näher vorgestellt.

Abbildung 3.12:
Datumsangaben in
deutscher Sprache

```
Morgen ist Samstag, der letzte Monat war ein Juli.
```

Zeitfunktionen

Ganz ähnlich wie mit dem Datum kann auch mit der Zeit umgegangen werden. Tabelle 3.13 zeigt die entsprechenden Funktionen.

Tabelle 3.13:
Zeitfunktionen

Funktion	Beschreibung
<code>mktime</code>	Ermittelt den Zeitstempel für eine bestimmte Zeitangabe.
<code>gmmktime</code>	Wie <code>mktime</code> , aber für GMT (Greenwich Mean Time).
<code>time</code>	Gibt den Unix-Zeitstempel sekundengenau zurück.
<code>microtime</code>	Wie <code>time</code> , aber mit der Genauigkeit Microsekunde (ist nicht unter Windows verfügbar).
<code>strftime</code>	Formatiert eine Zeitangabe.

Funktion	Beschreibung
gettimeofday	Gibt die Tageszeit zurück.

Viele Datums- und Zeitfunktionen rechnen mit der internen Angabe des Unix-Zeitstempels. Diese Angabe stellt die Anzahl der Sekunden seit dem 1.1.1970 00:00 Uhr dar. Um nun eine solche Angabe für ein spezifisches Datum zu erhalten, setzen Sie `mktime` ein:

```
$stamp = mktime(stunde, minute, sekunde, monat, tag, jahr, sz);
```

Der Parameter `sz` ist optional. Wird er auf 1 gesetzt, nimmt die Funktion an, dass sich das Datum in der Sommerzeit befindet. Der Wert ist 0 außerhalb der Sommerzeit, der Standardwert ist -1 (unbekannt).

Der Zeitstempel für den Frühlingsanfang 2002 wird also folgendermaßen ermittelt:

```
$stamp = mktime(0,0,0,3,21,2002);
```

Formatfunktionen

Die Funktion `strftime` arbeitet ähnlich `date` und formatiert eine Datums- und Zeitangabe anhand einer Formatieranweisung.

Beachten Sie, dass die `strftime`-Parameter nur teilweise mit `date` übereinstimmen und in einigen Fällen eine völlig andere Bedeutung haben.



Die Parameter werden dabei immer von einem Prozentzeichen eingeleitet und gelten entsprechend folgender Liste:

- %a
Abkürzung des Wochentags: »Mon«.
- %A
Voller Name des Wochentags: »Monday«.
- %b
Abkürzung des Monats: »Jan«.
- %B
Monat, ausgeschrieben: »January«.
- %c
Vollständige Datums- und Zeitangabe entsprechend den lokalen Einstellungen (siehe `setlocale` weiter unten).
- %C
Jahrhundertwert, wobei für 2002 »20« zurückgegeben wird.

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

- %d
Tag des Monats mit führender Null: »01« bis »31«.
- %H
Stunde im 24-Stunden-Format, »00« bis »23«.
- %I
Stunde im 12-Stunden-Format: »01« bis »12«.
- %j
Tag im Jahr: »001« bis »366«.
- %m
Monat ohne führende Null: »1« bis »12«.
- %M
Minuten als Ziffer: »0« bis »59«.
- %n
Fügt einen Zeilenvorschub ein (wie sonst mit \n).
- %p
»am« oder »pm«.
- %r
Komplette Zeit mit »am« oder »pm«.
- %S
Sekunden als Ziffer: »0« bis »59«.
- %T
Komplette Zeit in der Form HH:MM:SS.
- %U
Wochennummer im Jahr, startet die Zählung am ersten Sonntag.
- %W
Wochennummer im Jahr, startet die Zählung am ersten Montag.
- %w
Numerische Darstellung des Wochentags: »0« (für Sonntag) bis »6« (Samstag).

- %x
Vollständige Datumsangabe entsprechend den lokalen Einstellungen (siehe `setlocale` weiter unten).
- %X
Vollständige Zeitangabe entsprechend den lokalen Einstellungen (siehe `setlocale` weiter unten).
- %y
Zweistellige Ausgabe des Jahres: »99« oder »00«.
- %Y
Vierstellige Ausgabe des Jahres: »1999« oder »2000«.
- %Z
Differenz zur Zeitzone in Sekunden: »-43 200« bis »43 200«. Dies entspricht -12 oder +12 Stunden.
- %%
Das Prozentzeichen.

Interessant ist an dieser Funktion, dass sich die Ausgaben mit Hilfe der Funktion `setlocale` an die sprachlichen Besonderheiten eines bestimmten Gebiets anpassen lassen. **setlocale()**

```
<?php
print(strftime("%A heißt auf Deutsch "));
setlocale ("LC_TIME", "ge");
print(strftime("%A.<br>"));
?>
```

*Listing 3.49:
strftime formatiert
Datumsangaben auch
in deutsch*

Sie sehen an dem Beispiel, dass der zweite Parameter entfallen kann, auch hier wird wieder die aktuelle Zeit des Servers eingesetzt. Die Funktion `setlocale` selbst hat folgenden Aufbau:

```
string setlocale(category, localeID);
```

Der Parameter *category* wird durch folgende Werte bestimmt:

- LC_ALL
Alle weiteren Angaben.
- LC_COLLATE
Wirkt auf Zeichenkettenvergleiche.
- LC_CTYPE
Wirkt auf die Zeichensetzung, beispielsweise in `strtoupper()`.

- LC_MONETARY

Wirkt auf Währungsfunktionen wie `localeconv()`.

- LC_NUMERIC

Bestimmt das Dezimaltrennzeichen.

- LC_TIME

Wirkt auf Datums- und Zeitformatierungen mit `strftime()`.

Als *localeID* wird der ISO-Landescode angegeben, auf dessen Parameter die Ausgaben gesetzt werden sollen. Wird eine leere Zeichenkette genutzt, versucht PHP entsprechende (gleichlautende) Variablen in der Betriebssystem-Umgebung zu finden. Wird eine »0« angegeben, werden die aktuellen Einstellungen nicht geändert, aber zurückgegeben. Die Funktion gibt `FALSE` zurück, wenn der lokale Code nicht implementiert wurde.



Die Ländercodes der ISO-Tabelle berücksichtigen sowohl die Sprachen als auch das Land. Da heißt es dann für Deutschland »de_DE«, für Schweiz gibt es zwei Varianten: »de_CH« und »fr_CH«. Die konkrete Umsetzung hängt jedoch vom Betriebssystem ab. Unix liefert diese Tabellen in `/usr/share/language` mit. Je nach System gelten die dort befindlichen Daten. Für Windows liefert PHP hire eigene Definitionen. Beim Provider werden sie nicht umhin kommen, dies auszuprobieren.

Interessant an den lokalisierten Einstellungen ist die Auswirkung auf die interne Verarbeitung von Zahlen. Normalerweise gilt für alle numerischen Werte die Schreibweise mit Dezimalpunkt. Daran ändern auch die lokalen Einstellungen mit `setlocale` nichts. Wenn man dagegen Zahlen als Zeichenkette angibt und dann mit diesen einfache mathematische Operationen ausführt, rechnet PHP auch mit Dezimalkommata. Der folgende Code zeigt, wie das aussieht:

Listing 3.50:
Auswirkungen der
Einstellung mit
setlocale auf
Zahlen: setlocale

```
<?
setlocale("LC_ALL", "ge");
$net = "1234,56";
$gross = "1,16" * $net;
printf("Netto: %04.2f, Brutto: %04.2f", $net, $gross);
?>
```

Wie es funktioniert

Die Funktion basiert auf der internen Umwandlung zwischen Datentypen, die PHP automatisch vornimmt. Da der Operator `*` auf Zeichenketten keine Anwendung finden kann, wird der passende Datentyp (in diesem Fall `double`) ermittelt und dann die Multiplikation ausgeführt. Die folgende Zeile ist dagegen nicht zulässig:

```
$net = 1234,56;
```

Bevor Sie über den intensiven Einsatz der Funktion `setlocale` nachdenken, sollten Sie sich intensiv mit `number_format` beschäftigen. Die Formatierung zur Ausgabe birgt deutlich weniger Risiken und ist weitestgehend versions- und plattformunabhängig. Das Beispiel in Listing 3.50 zeigt nämlich außerdem, dass `setlocale` nicht auf `printf` wirkt, was eigentlich nicht verständlich ist (zumindest war dies bis PHP 4.0.5 der Fall).

```
Netto: 1234.56, Brutto: 1432.09
```

Abbildung 3.13:
Ausgabe von Listing
3.50

Tipps für Formatierungen

Es ist eine gute Idee, alle internen Berechnungen und Verarbeitungen mit den Standarddatenformaten, also englischen Zahlen und Daten im Unix-Timestamp-Format vorzunehmen und ausschließlich zur Ausgabe umzuwandeln. Dies reduziert die Zahl der Eingriffe bei der Übertragung des Skripts von einem System auf ein anderes erheblich.

3.2.12 Reguläre Ausdrücke

Reguläre Ausdrücke sind ein besonderes Kapitel wert. Dieser Teil beschreibt eine Einführung, um den ersten Kontakt zu erleichtern. Weitere Informationen finden Sie in ➤Abschnitt 9.5 *Reguläre Ausdrücke* ab Seite 668, dort für fortgeschrittenere Programmierer aufbereitet.

In einem regulären Ausdruck steckt, wenn man es genau betrachtet, ein komplettes Programm. Wenn man dies akzeptiert und versteht, dann ist der Entwurf und die Nutzung nicht schwieriger als mit jedem anderen Stück Code auch. Sie müssen sich die Zeit nehmen, einen regulären Ausdruck sorgfältig zu entwerfen und zu testen. Niemand – auch Profis nicht – schreiben einen neuen Ausdruck, der vielleicht nur 10 oder 20 Zeichen umfasst, in ein paar Sekunden auf. Intern arbeitet auch nicht ein einfacher Mustervergleich, sondern die Regex-Maschine, ein Programm, das die Ausdrücke auflöst und ausführt. Das wird bei umfangreichen Konstrukten spürbar – die Ausführung einer Suche mit einem regulären Ausdruck kann signifikant Rechenzeit in Anspruch nehmen. Und nicht zuletzt können solche Gebilde auch zu Programmschleifen führen, Abstürze verursachen und sich sonst wie jedes andere Programm daneben benehmen.

**Wie der Ausdruck
intern arbeitet**

Es ist eine Kunst, gute Programme zu schreiben. Ebenso ist es eine Kunst, gute reguläre Ausdrücke zu schreiben. Tatsächlich kann man komplexe Aussagen oft auf mehr als einem Weg ausdrücken. Der eine Weg ist vielleicht schneller, der andere kompakter, der dritte verständlicher. Lassen Sie sich nicht davon abhalten, reguläre Ausdrücke einzusetzen. So wie Ihre Programme im Laufe der Zeit immer besser werden, optimieren Sie auch die Suchmuster.

**Reguläre
Ausdrücke sind
nicht trivial**



Eine kompakte Einführung

Reguläre Ausdrücke (engl. *Regular expressions*) beschreiben Suchmuster. Diese Suchmuster können so komplex sein, dass sich ganze Filter in einer Zeile beschreiben lassen. Durch Schalter und Funktionen werden diese Muster dann zu einem mächtigen Werkzeug. Die einfachste Form, ein Suchmuster anzugeben, besteht in der Klammerung einer Zeichenkette mit Anführungszeichen:

"Suchwort"

^
\$

Die Steuerung innerhalb des Suchmusters wird durch spezielle Schalter vorgenommen – Symbole, die eine besondere Bedeutung haben. Die wichtigsten Symbole sind:

- ^

Kennzeichnet den Beginn der Zeichenkette

- \$

Kennzeichnet das Ende der Zeichenkette

Mit diesen Zeichen kann die Suchfunktion erkennen, wo die zu suchende (oder zu ersetzende) Zeichenkette beginnt und endet. Einige Beispiele zeigen, wie das zu verstehen ist:

- "^Skript"

Mit der Kennzeichnung wird erreicht, dass alle Zeichenketten erkannt werden, die mit den Zeichen »Skript« beginnen; ^ steht für den Wortanfang. Beispielsweise werden folgende Zeichenfolgen erkannt:

- »Skript«, »Skripts«, »Skriptsprache«, »Skriptprogrammierer«
- Nicht erkannt wird dagegen: »PHP-Skript«

- "sprach\$"

Hier werden alle Zeichenketten erkannt, die auf die Zeichen »sprach« enden. Erkannt werden:

- »Skriptsprache«, »Programmiersprache«
- Nicht erkannt werden dagegen: »Die Sprache«, »Sprachschatz«

- "^PHP\$"

Diese Angabe verlangt, dass die Zeichenkette sowohl den Wortanfang als auch das Wortende erfüllt, es kann also nur eben diese Zeichenfolge selbst passen: »PHP«.

- "ript"

Hier wird die Zeichenfolge erkannt, egal wo sie sich im Suchwort befindet. Erkannt werden:

- »Skript«, »Skriptsprache«, »script«, »Skriptprogrammierer«

Die Symbole *, + und ? kennzeichnen Wiederholungen von alphanumerischen Zeichen. Sie haben folgende Bedeutung:

- *

Keines oder beliebig viele Zeichen; der klassische Platzhalter⁹.

- +

Ein oder mehr Zeichen.

- ?

Kein oder genau ein Zeichen; auch ein üblicher Platzhalter.

Hier einige Beispiele, wie sich die Symbole verhalten:

- "12*"

Hier passen Zeichenketten, die eine 1 und eine beliebige Anzahl 2 enthalten, also »1«, »12«, »122«, »1222« usw.

- "12+"

Hier muss mindestens eine 2 vorhanden sein, also »12«, »122«, »1222« usw., nicht jedoch »1«.

- "12?"

Hier muss genau eine oder keine 2 vorhanden sein, also nur »1« oder »12« erfüllen diese Bedingung.

- "1?2+\$"

Eine etwas anspruchsvollere Kombination. Mögliche Übereinstimmungen ergeben sich mit »12«, »2«, »122« oder »2222«; wobei dies jeweils am Ende der zu durchsuchenden Zeichenkette stehen muss.

Die Wiederholung kann noch feiner gesteuert werden, wenn geschweifte Klammern gesetzt werden. Die folgenden Beispiele zeigen die Anwendung:

⁹ Hier und im Folgenden wird die deutsche Bezeichnung Platzhalter verwendet. Üblich sind auch Namen wie Joker oder Wildcard.

- "DL{2}"

Dies entspricht exakt »DLL«, das »D« muss einmal vorkommen, gefolgt von 2 »L«.

- "DL{2,}"

Mit dem Komma wird der Operator auf »mindestens« erweitert, gültige Übereinstimmungen sind »DLL«, »DLLLL« usw.

- "DL{2,3}"

Der zweite Parameter des Klammeroperators setzt die obere Grenze. Dieses Beispiel erfüllt nur die Varianten »DLL« und »DLLL«.

Bei den Klammern können Sie die obere Grenze weglassen, nicht jedoch die untere. Allerdings kann 0 eingesetzt werden. {0,3} ist korrekt, während {,3} nicht zulässig ist.

Es gibt einige Ersetzungen, die das Lesen von regulären Ausdrücken nicht unbedingt erleichtern. Es ist durchaus möglich, dass zwei völlig verschiedene Ausdrücke denselben Zweck erfüllen. Die folgende Auflistung zeigt, welche Ausdrücke übereinstimmen:

- * entspricht {0,}
- + entspricht {1,}
- ? entspricht {0,1}

()

Nun sind die bisher gezeigten Ausdrücke wenig praxisnah. Normalerweise wird eher nach ganzen Wörtern gesucht als nach einzelnen Zeichen. Alle Operatoren lassen sich natürlich auch auf Wörter oder Zeichengruppen anwenden. Dazu werden diese Gruppen in runde Klammern gesetzt:

- "1(34)*"

Hier kann sich die Folge »34« beliebig oft wiederholen, gültig wären: »1«, »134«, »13434« usw.

- "1(34){1,2}"

Hier darf die »34« nur ein- oder zweimal auftreten: »134« und »13434« erfüllen die Bedingung.

|

Oft sind verschiedene Schreibweisen zu berücksichtigen. In regulären Ausdrücken sind deshalb weitere Operatoren zulässig. Das Symbol senkrechter Strich (|) dient als Oder-Operator:

- `"Dr|Prof"`

Hier wird die Übereinstimmung mit »Dr« oder mit »Prof« erkannt. Die Position in der durchsuchten Zeichenkette spielt keine Rolle, so erfüllt auch der Name »Dreher« die Bedingung.

- `"^(Dr|Prof)\."`

Diese Form erfüllt den Zweck besser. Der Punkt ist aber auch ein Sonderzeichen und wird mit einem Backslash zum »normalen« Zeichen gemacht. Erkannt werden nun »Dr.« und »Prof.«, wenn sie am Anfang der Zeichenkette stehen.

- `"(1|2|3)*0"`

Dies ist wieder eine etwas komplexere Form. Erkannt werden alle Zeichenfolgen mit einer »0« am Ende, davor dürfen, beliebig gemischt und beliebig oft, die Zeichen »1«, »2« und »3« auftreten. Erkannt werden: »1230«, »10«, »2220«, »3210« usw.

Dass der Punkt `.` eine Sonderrolle spielt, wurde bereits angesprochen. Er steht einfach für genau ein beliebiges Zeichen. Die Anwendung ist vielfältig, wenn er mit anderen Zeichen kombiniert wird. Eine Auswahl aus einer Folge von Zeichen kann mit eckigen Klammern erfolgen, wie die folgenden Beispiele zeigen:

- `"^.{5}"`

Diese Symbole stehen für genau 5 beliebige Zeichen.

- `".[0-9]"`

Hier muss ein beliebiges Zeichen von genau einer Ziffer gefolgt werden, also »A0«, »x7«, »R9« usw.

Dabei stehen die Zeichen in der eckigen Klammer immer für genau ein Zeichen. Im letzten Beispiel wäre auch »23« gültig, wenn die Folge aber mit einem Buchstaben beginnen soll, schreiben Sie:

- `"([a-z]|[A-Z])[0-9]"`

Dies sollte noch keine Probleme bereiten. Gültig sind beispielsweise: »A3«, »f7«, »c9«, nicht aber »AA« oder »26«.

Die Zeichenfolge `[a-z]|[A-Z]` kann auch kürzer geschrieben werden: `[a-zA-Z]`. Einige Beispiele:

- `"[a-f0-9]"`

Dies ist gut geeignet, um Hexadezimalzahlen zu erkennen.

- `"^[a-zA-Z]"`

Die Zeichenfolge muss mit einem Buchstaben beginnen.

- "[0-9]%".

Vor einem Prozentzeichen muss eine Ziffer stehen.

- "[0-9],[0.9]"

Vor und nach einem Komma müssen Ziffern stehen.

Innerhalb der eckigen Klammern hat das ^-Zeichen eine besondere Bedeutung: es negiert die Auswahl. Wollen Sie alle Zeichen außer Buchstaben ausschließen, wäre diese Form zu verwenden:

- "[^a-zA-Z]"

Einsatz der Auswahlnegation.

\ (Backslash)

Die bisher eingeführten Sonderzeichen können natürlich in normalen Zeichenketten auftreten. Sie müssen dann mit dem Backslash \ gekennzeichnet werden. Hier eine Zusammenfassung aller speziellen Zeichen:

```
^ . [ ] $ ( ) | * + ? { } \
```

Für den Backslash selbst schreiben Sie also \\. Wichtig ist in diesem Zusammenhang die Reaktion der Zeichen auf die eckigen Klammern. Hier verlieren alle Sonderzeichen ihre spezielle Bedeutung:

- "[^?+\|]"

Diese Form bringt eine Übereinstimmung, wenn eines der Zeichen ^ ? + \ | auftritt.

Erweiterte Syntax

Reguläre Ausdrücke in PHP arbeiten nach dem POSIX-Standard 1003.2. Dieser Standard erlaubt auch die Angabe generischer Zeichenklassen innerhalb der Klassendefinition [].

:symbol:

Einige besonders häufig benötigte Zeichenfolgen haben ein spezielles Symbol in der Form :symbol:. Tabelle 3.14 zeigt alle Möglichkeiten.

Tabelle 3.14:
POSIX-Ersatz-
symbole für reguläre
Ausdrücke

POSIX-Ersatzsymbol	Bedeutung
:alnum:	Alphanumerische Zeichen
:alpha:	Zeichen des Alphabets
:digit:	Numerische Zeichen (Ziffern, Plus, Minus)
:blank:	Leerzeichen und Tabulator
:space:	Alle »weißen« Zeichen
:punct:	Satzzeichen
:lower:	Kleinbuchstaben

POSIX-Ersatzsymbol	Bedeutung
:upper:	Großbuchstaben
:cntrl:	Steuerzeichen
:graph:	Druckbare und sichtbare Zeichen
:print:	Alphanumerische Zeichen
:xdigit:	Hexadezimale Zeichen (0-9,a-f,A-F)

Einsetzen können Sie diese Symbole in eckigen Klammern:

- "[a-zA-Z0-9]" entspricht "[a1num:]"

Neben dem Suchen dienen reguläre Ausdrücke oft auch zum Ersetzen von Teilen von Zeichenketten. Auch bei der Angabe der Ersatzzeichen sind einige Sonderzeichen möglich:

- \t steht für den Tabulator
- \n für einen Zeilenumbruch (newline)

Beispiele für reguläre Ausdrücke

Der Sinn und praktische Einsatz regulärer Ausdrücke mag sich nach der ersten Vorstellung der Syntax noch nicht jedem Leser erschließen. Nachfolgend werden einige Beispiele vorgestellt, die bei der Skripterstellung eingesetzt werden können.

Auf die konkrete Umsetzung in PHP und die Funktionen für reguläre Ausdrücke wird im Anschluss an diese Einführung eingegangen.



Eingabefelder in HTML kennen als Format nur Zeichenketten. Es obliegt dem Programmierer, die eingetragenen Werte auf korrekte Schreibweise hin zu untersuchen. Reguläre Ausdrücke erledigen dies in einer einzigen Programmzeile. Diese können Sie übrigens nicht nur in PHP einsetzen. Auch in JavaScript sind, wie in vielen anderen Programmiersprachen, reguläre Ausdrücke verfügbar. Der Einarbeitungsaufwand lohnt sich also unbedingt. Die Syntax der Ausdrücke ist weitgehend gleich, lediglich die aufrufenden Funktionen unterscheiden sich. Auch dies ist eine Besonderheit regulärer Ausdrücke: sie stehen in fast allen Programmiersprachen zur Verfügung.

Eingabefelder untersuchen

Auf die Auswertung von Formularen wird in Kapitel 4 ausführlich eingegangen (siehe dazu Abschnitt 4.1 *Formulare auswerten* ab Seite 301).



Wenn Sie Geldwerte erwarten, können diese unterschiedlich eingegeben werden; DM 1.522 können auch »1522«, »1.522«, »1522,00« oder »1.522,00« sein. Die Entwicklung des passenden regulären Ausdrucks soll hier Schritt für Schritt vorgestellt werden.

Geldwerte

- $^{\wedge}[1-9][0-9]^*\$$

Dieser Ausdruck akzeptiert nur Ziffern und Ziffernfolgen. Das erste Zeichen darf nicht 0 sein.

- $^{\wedge}(0|[1-9][0-9]^*)\$$

Dieser Ausdruck akzeptiert nur Ziffern und Ziffernfolgen. Das erste Zeichen darf nicht 0 sein, eine einzelne 0 wird aber akzeptiert, was bei Geldwerten durchaus möglich ist.

- $^{\wedge}(0|-?[1-9][0-9]^*)\$$

Nun wird zusätzlich noch ein Minuszeichen erlaubt, wenn der Wert nicht 0 ist. Minuszeichen werden aber oft gar nicht sinnvoll sein, dafür fehlen noch die Dezimalstellen.

- $^{\wedge}[0-9]+(,[0-9]+)\$$

Hier werden ein oder mehr Ziffern vor und nach dem Komma akzeptiert.

- $^{\wedge}[0-9]+(,[0-9]\{2\})\$$

Eine Verbesserung: genau zwei Nachkommastellen sind erlaubt.

- $^{\wedge}[0-9]+(,[0-9]\{1,2\})\$$

Noch eine Verbesserung: entweder ohne Komma oder mit Komma und ein oder zwei Nachkommastellen.

- $^{\wedge}[0-9]\{1,3\}(\.[0-9]\{3\})^*(,[0-9]\{1,2\})?\$$

Hier wird der Tausenderpunkt akzeptiert. Zahlen wie »1.555« werden ebenso akzeptiert wie »34,55«. Richtig nutzbar wird erst die letzte Version:

- $^{\wedge}[0-9]+|[0-9]\{1,3\}(\.[0-9]\{3\})^*(,[0-9]\{1,2\})?\$$

Das war einfach! Wie viele Schleifen hätten Sie bei der herkömmlichen Programmierung benötigt?

Bei der weiteren Auswertung ist zu beachten, dass die Untersuchung einer Zeichenkette mit Hilfe regulärer Ausdrücke noch keine Bewertung darstellt.



Hinweis

E-Mail-Adresse untersuchen

Bevor Sie die Zeichenkette für mathematische Operationen nutzen, entfernen Sie die Tausenderpunkte und tauschen die Kommata gegen Punkte aus: PHP rechnet intern mit amerikanischen Zahlenformaten.

Eine andere Form der Eingabe tritt auch häufiger auf: die E-Mail-Adresse. Hier ist es sicher möglich, mit Hilfe von Anfragen an Name-server die Existenz der Domain zu testen und andere Maßnahmen zu ergreifen, um richtige von falschen Mailadressen zu unterscheiden.

Mit einem regulären Ausdruck können Sie zumindest die grundsätzliche Form überprüfen.

Ein POP3-Mailname ist aus kleinen und großen Buchstaben aufgebaut. Als zusätzliche Zeichen sind das Minuszeichen »-«, der Unterstrich »_« und der Punkt ».« erlaubt. Groß- und Kleinschreibung spielt keine Rolle. Nach dem Namen folgt das @-Zeichen und die Domain. Hier dürfen keine Unterstriche auftreten, sonst entspricht es dem Mailnamen. Der reguläre Ausdruck wird wieder Schritt für Schritt aufgebaut.

- `^[_a-zA-Z0-9-]+$`

Dieser Teil untersucht den Mailnamen auf gültige Zeichen. Der Punkt fehlt noch.

- `^[_a-zA-Z0-9-]+(\.[_a-zA-Z0-9-]+)*$`

Dieser Ausdruck akzeptiert auch den Punkt, allerdings nicht am Anfang oder am Ende des Namens.

- `^[a-zA-Z0-9-]+\.[a-zA-Z]{2,3}$`

Hier wird nur der Domainname untersucht. Unterstriche sind nicht erlaubt, der Punkt ist zwingend und wird von 2 oder 3 Zeichen gefolgt, die wiederum nur Buchstaben sein können.

- `^[_a-zA-Z0-9-]+(\.[_a-zA-Z0-9-]+)*@[a-zA-Z0-9-]+\.[a-zA-Z]{2,3}$`

Das Gebilde ist nun schon recht unübersichtlich. Es ist eine Kombination aus dem vorderen und dem hinteren Teil, verbunden durch das @-Zeichen.

Um die Mailfunktion einsetzen zu können, ist noch eine weitere Verfeinerung nötig. Die Angabe von mehrfachen Domainnamen (beispielsweise »mailadresse@mail.subdomain.domain.de«) ist noch nicht möglich.

- `([a-zA-Z0-9-]+\.)+`

Der erste Teil des zweiten Ausdrucks kann deshalb mehrfach wiederholt werden, muss aber mindestens einmal erscheinen. Eine weitere Klammer mit einem +-Zeichen erledigt das zufriedenstellend.

Das Ergebnis noch einmal in voller Pracht; diese Schreibweise eignet sich übrigens auch gut zur Fehlersuche:

```
^
  [_a-zA-Z0-9-]+
  (\.
    [_a-zA-Z0-9-]+
  )*
  @
```

```
(
    [a-zA-Z0-9-]+
    \.
)+
(
    [a-zA-Z]{2,3}
)
$
```

Die klassischen Funktionen

In PHP 3 wurden die Funktionen `ereg`, `ereg_replace` usw. zur Verarbeitung regulärer Ausdrücke eingeführt. Diese sind zwar immer noch verfügbar, sollten aber nicht verwendet werden. Die Ausdruckssprache realisiert nur einen Teil der aus Perl bekannten Syntax und die Geschwindigkeit der Regex-Maschine ist mangelhaft. Ganz bewusst werden diese Funktionen deshalb hier »ausgesperrt«. Verwenden Sie die komfortableren *preg*-Funktionen, die im folgenden Abschnitt angesprochen werden.

Funktionen für reguläre Ausdrücke

Die folgenden Funktionen verarbeiten reguläre Ausdrücke und bringen die gezeigten Konstrukte zur Anwendung im PHP-Skript:

- `preg_match`
Suche nach einem Vorkommen des Musters.
- `preg_match_all`
Suche nach allen Vorkommen des Musters.
- `preg_replace`
Suchen und Ersetzen aller Vorkommen.
- `preg_split`
Zerlegen einer Zeichenkette in ein Array anhand eines Suchmusters.

Weitere Funktionen finden Sie im Abschnitt 9.5 *Reguläre Ausdrücke* ab Seite 668, wo dasselbe Thema nochmals tiefergehend behandelt wird.

Die Funktionen verlangen den Einbau des Suchmusters in zwei Begrenzungszeichen. Dies kann jedes Zeichen sein – jedoch keine Buchstaben oder Zahlen und kein Zeichen, das bereits im Ausdruck selbst vorkommt.

Die bereits vorgestellten Beispiele im Einführungsteil waren verhältnismäßig einfach (auch wenn Leser, die noch nie mit regulären Ausdrücken zu tun hatten, dies anders empfinden). Bei der Suche nach

Übereinstimmungen können mehrere Treffer zurückgegeben werden. In den folgenden Beispielen wird als Argument daher oft ein Array übergeben, in dem dann mehrere Ergebnisse liegen können.

Die Funktion `preg_match` untersucht eine Zeichenkette anhand eines regulären Ausdrucks und füllt ein übergebenes Array mit Fundstellen. Falls die Suche erfolgreich war, wird `TRUE` zurückgegeben:

```
<?php
$date = "1999-07-16";
echo "Datum zu prüfen: $date<br>";
$preg = "([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})";
if (preg_match("/$preg/", $date, $regs)) {
    echo "$regs[3].$regs[2].$regs[1]";
} else {
    echo "Falsches Datumsformat: $date";
}
?>
```

Listing 3.51:
regex preg: Reguläre
Ausdrücke anwenden

Das Beispiel in Listing 3.51 zeigt die Umwandlung des amerikanischen Datumsformats in das deutsche. Die Platzierung der drei Werte in das Array wird durch die Klammerung der Ausdrucksteile erreicht. Ginge es nur um eine Ja/Nein-Entscheidung, ob das Format korrekt ist, könnten alle runden Klammern entfallen. Die zweite Funktion, die häufig eingesetzt wird, ist `preg_match_all`. Damit werden alle Vorkommen des Suchmusters in der Zeichenkette gefunden. Der Rückgabewert (dritter Parameter) ist dann immer ein Array.

Die Funktionen können durch Schalter ergänzt werden, die das Verhalten modifizieren. Die Schalter stehen nach dem letzten Begrenzungszeichen. Interessant ist »i«. Damit wird die Unterscheidung von Groß- und Kleinschreibung ausgeschaltet:

```
"/muster/i"
```

Die Suche allein ist meist nicht der entscheidende Punkt, sondern die Modifikation der Zeichenketten. Sie können mit der Funktion `preg_replace` die Fundstellen durch andere Zeichen ersetzen.

```
<?php
$email = "test#domain.de";
echo preg_replace("/#/","@", $email);
?>
```

Schalter

preg_replace()

Listing 3.52:
regex eregreplace:
Ersetzen nach
Suchmustern

Hier wird die neue (ersetzte) Zeichenkette zurückgegeben. Wenn der reguläre Ausdruck keine Übereinstimmung ergab, wird die originale Zeichenkette übergeben. `preg_replace` entspricht dem, nur wird die Groß- und Kleinschreibung generell nicht beachtet. Die Umwandlung wird auch für Umlaute korrekt ausgeführt.

Das einfache Beispiel in Listing 3.52 legt nahe, die Textersetzungsfunktion `str_replace` zu nutzen. Ein einfacher Test zeigt zudem, dass Funktio-

nen mit regulären Ausdrücken verhältnismäßig langsam sind. Hinter der Funktion `preg_replace` muss also mehr stecken. Tatsächlich ist die Anwendung regulärer Ausdrücke zum Ersetzen noch weitaus komplexer, als bisher beschrieben. Um dieses Kapitel nicht zu überfrachten und vor allem nicht abzuschrecken, wurde der weiterführende Teil zu regulären Ausdrücken in ➡ Kapitel 9 *Fortgeschrittene Programmierung* verbannt. Lesen Sie dort weiter, wenn Sie sich mit der hier beschriebenen Technik angefreundet und ein paar eigene Muster erfolgreich erstellt haben.

preg_split()

Die Funktion `preg_split` teilt eine Zeichenkette in Segmente und legt jedes Segment als Element in einem Array ab. Als Suchwort wird dabei ein regulärer Ausdruck akzeptiert.

*Listing 3.53:
regex_pregsplit:
Zeichenketten mit
split zerlegen*

```
<?php
$email = "name@domain.de";
$test = preg_split("/@/", $email);
echo "Name: $test[0]<BR>";
echo "Domain: $test[1]<BR>";
?>
```

Wird keine Übereinstimmung gefunden, steht die gesamte Zeichenkette anschließend im Array im Index 0.

split() spliti()

Alternativ kann auch die Funktion `split` verwendet werden, die jedoch auf dem leistungsschwächeren Regex-Modul von PHP basiert, nicht auf dem von Perl. Hier sind die Begrenzungssymbole nicht nötig und Optionen nicht zulässig. `spliti` unterdrückt die Berücksichtigung von Groß- und Kleinschreibung. Die Funktion `split` kennt noch einen dritten Parameter, der die Anzahl der Trennstellen begrenzt. Die Angabe ist optional – standardmäßig wird die gesamte Zeichenkette durchsucht.

3.2.13 Webspezifische Funktionen

Mit den webspezifischen Funktionen können Sie mit den speziellen Bedingungen im Internet umgehen. Die ersten drei Funktionen behandeln den URL (*uniform resource locator*).

parse_url()

Die Funktion `parse_url` übernimmt als Argument ein URL und erzeugt ein assoziatives Array mit den in der folgenden Übersicht gezeigten Schlüsseln. Betrachten Sie den folgenden URL:

```
http://www.test.de:8080/scripts/script.php?query=value&action=no
```

Daraus ergeben sich für die einzelnen Elemente des Arrays folgende Inhalte:

- `scheme`: http
- `host`: www.test.de

- port: 8080
- path: /scripts/script.php
- query: query=value&action=no

Die Ausgabe erzeugt das folgende Skript:

```
<?php
$url = "http://www.test.de:8080/scripts/script.php?query=value";
$array = parse_url($url);
echo $url,"<P>";
while(list($key, $value) = each($array))
    { echo "$key: $value <br>"; }
?>
```

```
http://www.test.de:8080/scripts/script.php?query=value
scheme: http
host: www.test.de
port: 8080
path: /scripts/script.php
query: query=value
```

Listing 3.54:
web_parseurl:
Analyse eines URL
mit parse_url

Abbildung 3.14:
Ausgabe der Details
eines URL (aus
Listing 3.54)

Die Funktionen `urldecode` und `urlencode` behandeln alle nicht alphanumerischen Zeichen in einem URL. Mit `urlencode` ersetzen Sie diese Zeichen durch die Zeichenfolge `%HH`, wobei `HH` den Hexcode des betreffenden Zeichens im ASCII-Zeichensatz darstellt. `urldecode` wandelt eine so codierte URL wieder zurück.

`urldecode()`
`urlencode()`

Die Funktion `base64_encode` verwandelt die als Parameter übergebenen Werte in 7-Bit-Code. Einige Mailprogramme verwenden 7-Bit, so dass die Übertragung von Binärdaten (8-Bit) nicht möglich ist. Solche Daten werden in 7-Bit übertragen. Moderne Mail-Clients sollten die so versendete Mail automatisch dekodieren. Wenn Sie solche Clients selbst mit PHP schreiben, verwenden Sie die Funktion `base64_decode`.

`base64_decode()`
`base64_encode()`

Wenn Sie vom 8-Bit- in das 7-Bit-Format kodieren, vergrößert sich der Code um etwa 33%.



Hinweis

Mehr Informationen zur Base64-Kodierung finden Sie in der RFC 2045, Sektion 6.8. Das ist sicher ein sehr theoretisches Papier. Wenn Sie jedoch aktiv programmieren, ist die Kenntnis solcher Grundlagen unerlässlich. Als Anfänger müssen Sie sich damit noch nicht auseinandersetzen.

RFC 2045

HTTP-Funktionen

Die folgenden beiden Funktionen werden nur kurz erwähnt, in Kapitel 4 finden Sie ausführlichere Informationen dazu.

Mit der Funktion `header` senden Sie Informationen an den Browser, bevor der Seitentext beginnt. Notwendigerweise dürfen vor dem Header keine anderen Ausgaben erfolgen, auch keine Leerzeichen. Welche Angaben im Header möglich sind, wird in ➤ Abschnitt 9.1

`header()`
RFC 2068

Netzwerkprogrammierung ab Seite 631 diskutiert. Was generell erlaubt ist, zeigt die HTTP 1.1-Spezifikation in der RFC 2068. Ebenso werden dort alle Variationen im Umgang mit diesen Funktionen gezeigt. Eine einfache Anwendung – als erster Eindruck – ist die Umleitung auf eine andere Seite:

```
header("Location: http://www.php.net");
exit;
```

setcookie()

Die Funktion `setcookie`, setzt ein Cookie. Cookies werden noch im Zusammenhang mit der Sessionsteuerung behandelt. Die einfachste Anwendung setzt ein Cookie ohne weitere Parameter:

```
setcookie("CheckLogin", "Krause");
```

Der Abruf der Cookies erfolgt automatisch, alle erkannten Cookies werden in gleichnamige Variablen platziert. Die Abfrage lautet dann:

```
echo $CheckLogin;
```

Gelöscht wird das Cookie, indem es ohne Wert erneut gesetzt wird. Der Browser erkennt dies und trägt es aus der Cookie-Liste aus. Eine explizite Löschanweisung ist nicht vorgesehen. Dies ist also kein Mangel in PHP, sondern bedingt durch die Cookie-Definition.



Mehr Informationen zu Cookies finden Sie auch bei der Firma Netscape unter der folgenden Adresse:

```
http://www.netscape.com/newsref/std/cookie\_spec.html
```

3.3 Programmieren mit PHP

Bei umfangreichen Projekten benötigen Sie Kontrollstrukturen, die den Programmfluss steuern. Dieser Abschnitt zeigt alles, was Sie dazu wissen müssen.

3.3.1 Blöcke und Strukturen

{ }

PHP kennt den Begriff des Blocks. C-Programmierer werden das sofort verstehen, Syntax und Nutzung sind identisch. Blöcke werden durch Paare geschweifter Klammern gebildet:

Syntax

```
if($test) {
    //Blockanweisung
}
```

PHP kann so erkennen, welche Befehle zusammengehören. Vor allem bei der Vermischung von PHP und HTML müssen Sie Blöcke bilden, denn PHP beendet einen nicht explizit markierten Block am Ende des Skriptfragments. Der folgende Code funktioniert nicht:

```
<?php if($test) ?>
<P>HTML ist toll!</P>
```

Sie müssen hier kennzeichnen, dass der HTML-Code zu dem if-Befehl gehört:

```
<?php if($test) { ?>
<P>HTML ist toll!</P>
<?php } ?>
```

Ausnahmen

Zu einigen Befehlen gibt es eine alternative Syntax, die auf die Kennzeichnung der Blockstruktur mit Klammern verzichtet. In solchen Fällen werden die Blöcke mit einem Doppelpunkt eingeleitet und mit einem Schlüsselwort beendet.

3.3.2 Bedingungen

Bedingungen sind das wesentliche Element zur Programmsteuerung. Es gibt praktisch kein sinnvolles Programm, das völlig ohne die Steuerung mit Bedingungen auskommt. Das Programm fällt mit Hilfe der Bedingung eine einfache Entscheidung: Ja oder Nein. Entsprechend müssen Ausdrücke, die in den Bedingungsbefehlen eingesetzt werden, logische Ausdrücke sein. Das Ergebnis muss für PHP als Wahr (TRUE) oder Falsch (FALSE) interpretierbar sein.

Um Ausdrücke zu konstruieren, werden Operatoren benötigt. Man kann dabei logische Operatoren und Vergleichsoperatoren unterscheiden. Den eigentlichen Programmablauf steuern dann Befehle wie `if`, die nachfolgend vorgestellt werden. Auch die Befehle zur Schleifensteuerung nutzen logische Ausdrücke.

Logische Operatoren

In vielen Abfragen wird eine logische Entscheidung (Ja/Nein) verlangt. Mit speziellen Operatoren können Sie Ausdrücke verknüpfen. Das Ergebnis eines korrekten logischen Ausdrucks ist immer 0 (FALSE, Falsch) oder 1 (TRUE, Wahr). Die folgenden Ausdrücke zeigen die Anwendung:

```
$x and $y // ist wahr, wenn $x UND $y wahr ist
$x or $y  // ist wahr, wenn $x ODER $y wahr ist
$x xor $y // falsch, wenn beide gleich sind
$x && $y  // entspricht and
$x || $y  // entspricht or
!$x       // negiert den Wert FALSE=True, True=FALSE
```

and
or
xor
!
&&
||
Syntax

Zwischen der Form `&&` und `and` gibt es keinen Unterschied in der Ausführung der Operation. Bei der Notation sieht es anders aus. Die Dar-

stellung mit Schlüsselworten statt Operatorsymbolen spart Klammern. Der folgende Ausdruck muss mit dem Schlüsselwort geschrieben werden:

```
if ($a == 45 and $b == 23)
```

Wenn Sie die Symbole verwenden, sind zusätzliche Klammern nötig:

```
if (($a == 45) && ($b == 23))
```

Vergleichsoperatoren

Um komplexe logische Ausdrücke erstellen zu können, benötigt man auch Vergleichsoperatoren. Die folgende Tabelle gibt einen Überblick.

==, !=, ===, !==
<, >, <=, >=

Tabelle 3.15:
Vergleichs-
operatoren

Operatorsymbol	Bedeutung
==	Gleichheit
!=	Ungleichheit
===	Identität
!==	Nicht identisch
>	Größer als
<	Kleiner als
>=	Größer als oder gleich
<=	Kleiner als oder gleich



Neu in PHP 4 sind die Identitätsoperatoren === und !==. Damit wird nicht nur der Inhalt einer Variable auf Gleichheit bzw. Ungleichheit getestet, sondern auch der Datentyp. Durch die interne Typkonvertierung führen sonst Vergleiche zwischen Zeichenketten, die Zahlen enthalten, und den unmittelbar angegebenen Zahlen zu einer Übereinstimmung. Mit dem Identitätsoperator wäre das nicht der Fall.

Typische Probleme

Vergessen Sie nie, dass der Gleichheitsoperator == nicht dem Zuweisungsoperator = entspricht – ein wesentlicher Unterschied zu Sprachen wie VBScript, mit dem vor allem Umsteiger zu kämpfen haben. Der folgende Ausdruck ist syntaktisch völlig korrekt, trotzdem falsch (weil sinnlos):

```
if ($var = 17) {  
    echo $var;  
}
```

Gemeint war sicher, dass die Variable ausgegeben wird, wenn der Inhalt gleich 17 ist. Praktisch passiert aber Folgendes: Die Variable wird durch den Zuweisungsausdruck gleich 17 gesetzt – unabhängig von ihrem vorherigen Inhalt. Dann wird die 17 ausgegeben. Die Zuweisung selbst, als Ausdruck betrachtet, wird korrekt abgearbeitet

und von `if` als Wahr erkannt. Wenn Sie so einen bestimmten Zustand im Programm erkennen möchten, wird Ihnen dies vielleicht nicht einmal auffallen.

Es gibt eine einfache Lösung zum Verhindern solcher Fallen. Vertauschen Sie einfach die Operanden: **Trickreiche Lösung**

```
if (17 = $var)
```

Dieser Ausdruck ist syntaktisch definitiv falsch. Zuweisungen können nicht »rückwärts« arbeiten. Der Fehler wird vom Parser sofort erkannt. Schreiben Sie dagegen den Gleichheitsoperator korrekt, ergibt sich ein gültiger Ausdruck:

```
if (17 == $var)
```

Dies sieht zwar ungewöhnlich aus, ist aber sehr programmierfreundlich, zumindest für Anfänger. Profis nutzen das dagegen sehr selten. **Bei Profis selten zu sehen**

Bedingungen mit `if` auswerten

Die Syntax des Befehls `if` lehnt sich an C an. Entsprechend knapp fällt die Schreibweise aus: **if**

```
if ($tag == "Montag") echo "Heute ist Montag";
```

Der `if`-Befehl besteht aus dem Schlüsselwort und dem in runden Klammern stehenden logischen Ausdruck. Wenn der Ausdruck Wahr ist, wird der nachfolgende Befehl oder Block (in geschweiften Klammern) ausgeführt. Ist der Ausdruck Falsch, wird die Programmausführung mit dem nächsten Befehl oder dem nachfolgenden Block fortgesetzt. Folgen mehrere abhängige Befehle, ist also folgendermaßen zu schreiben:

```
if ($tag == "Montag") {  
    echo "Heute ist Montag";  
    echo "<P>";  
}
```

Das folgende Beispiel nutzt die schon bekannten Datumsfunktionen, um eine entsprechende Begrüßung zu erzeugen:

```
<?php $tag = date("l") ?>  
<?php if ($tag == "Monday") { ?>Heute ist Montag<?php }?>  
<?php if ($tag == "Tuesday") { ?>Heute ist Dienstag<?php }?>  
<?php if ($tag == "Wednesday") { ?>Heute ist Mittwoch<?php }?>  
<?php if ($tag == "Thursday") { ?>Heute ist Donnerstag<?php }?>  
<?php if ($tag == "Friday") { ?>Heute ist Freitag<?php }?>  
<?php if ($tag == "Saturday") { ?>Heute ist Samstag<?php }?>  
<?php if ($tag == "Sunday") { ?>Heute ist Sonntag<?php }?>
```

*Listing 3.55:
statement if:
Steuerung mit `if`*

Das Beispiel ist vergleichsweise primitiv. Sie können aber gut erkennen, wie Sie mit verflochtenem HTML-Code umgehen – jede Sequenz

wird von PHP als Block verstanden und *muss* in geschweifte Klammern gesetzt werden. Da dieser Fall relativ häufig auftritt, verfügt PHP hier über eine erweiterte Syntax, die praktisch eine Mischung aus Basic und C darstellt. Mehr dazu im ➡Abschnitt *Alternative Syntax* Seite 235.

if ist unbegrenzt verschachtelbar

Wenn man weiter zu komplexeren Entscheidungsbäumen vordringt, wären Verschachtelungen unvermeidlich. Prinzipiell kann der `if`-Befehl unbegrenzt verschachtelt werden, das heißt, in jedem Block kann sich wiederum ein `if`-Befehl verstecken. Das führt in aller Regel nicht zu besonders gut lesbarem Code und sollte vermieden werden. Es gibt fast immer elegantere Lösungsmöglichkeiten.

Oft ist es notwendig, nicht nur auf das Eintreten eines Ereignisses zu reagieren, sondern auch die negative Entscheidung zu behandeln. In solchen Fällen wird der `if`-Befehl um `else` ergänzt. Der Befehl oder Block hinter `else` wird ausgeführt, wenn die Bedingung *nicht* zutrifft.

**else
elseif**

Das folgende Beispiel führt eine solche zweistufige Entscheidung durch; egal wie das Datum ist, in jedem Fall muss eine Ausgabe erfolgen.

Listing 3.56:
*statement else: if in
Verbindung mit else*

```
<?php
$tag = date("w");
if ($tag == 0 or $tag == 7) {
    echo "Wochenende!";
} else {
    echo "Arbeitszeit";
}
?>
```

Das Beispiel aus Listing 3.55 könnte man auch mit `elseif` schreiben. Praktisch wird dabei in dem `else`-Zweig noch ein weiterer `if`-Befehl eingebaut. `elseif` spart also im Wesentlichen nur Schreibarbeit. Das folgende Beispiel zeigt eine mögliche Anwendung:

Listing 3.57:
*statement elseif:
elseif*

```
<?php
$tag = date("w");
if ($tag == 0 or $tag == 7) {
    echo "Wochenende!";
} elseif ($tag == 5) {
    echo "Fast Wochenende...";
} else {
    echo "Arbeitszeit";
}
?>
```

Die Funktion `date`, die hier eingesetzt wird, kann mit einem zweiten Parameter erweitert werden, der den Zeitwert bestimmt, für den das Datumsformat gilt. Zeitwerte in PHP entsprechen dem Unix-

Timestamp, also der Anzahl der Sekunden seit dem 1.1.1970. Das schränkt den Wertebereich natürlich etwas ein.

Wenn Sie die Beispiele für jeden Tag simulieren möchten, verwenden Sie die Funktion `mktime()` als zweiten Parameter in der Funktion `date()`. Mit `mktime(0,0,0,07,13,99)` simulieren Sie beispielsweise den 13. Juli 1999.



Hinweis

Alternative Syntax

Das Beispiel in Listing 3.55 zeigte eine große Schwierigkeit im Umgang mit verschachteltem HTML-Code. Die ursprünglich zugrunde gelegte C-Syntax ist hier wenig geeignet. PHP erweitert deshalb diese Schreibweise um eine eigene Alternative.

Mit der Einführung des Schlüsselwortes `endif` kann das Ende des Blocks explizit definiert werden. Die Klammern können dafür entfallen. Die `else-` und `elseif-`Befehle sind mit einem Doppelpunkt zu versehen, um die Zugehörigkeit zum Block zu kennzeichnen. Auch hier können die geschweiften Klammern entfallen.

```
<?php
$tag = date("w");
if ($tag == 0 or $tag == 7):
?>
    Wochenende!
<?php
    elseif ($tag == 5):
?>
    Fast Wochenende...
<?php else: ?>
    Arbeitszeit
<?php endif ?>
```

Listing 3.58:
statement_endif:
Alternative Syntax
mit `endif`

Kurzschreibweise

PHP kennt, wie C, eine Kurzschreibweise für einen einfachen Bedingungsbe- ... ? ... : ...; fehl, den sogenannten ternären Bedingungsoperator:

```
$var == "test" ? $result = TRUE : $result = FALSE;
```

Tatsächlich ist die gesamte Konstruktion ein Ausdruck. Ausdrücke geben immer ein Ergebnis zurück. Eleganter ist deshalb folgende Schreibweise:

```
$result = $var == "test" ? TRUE : FALSE;
```

Damit lassen sich Abfragen oft kürzer und lesbarer gestalten. Wenn Sie dagegen komplizierte Bedingungen entwerfen, leidet die Lesbarkeit wieder und die klassische Form mit `if` ist eher angebracht.



Es geht nicht darum, mit allen erdenklichen Tricks Programme auf so wenig Zeilen wie möglich zu schreiben, sondern lesbaren und übersichtlichen Code zu entwickeln. Kurzschreibweisen sind dabei nur selten ein probates Hilfsmittel.

Die drei Elemente haben folgende Bedeutung:

Logischer Ausdruck ? Wenn **TRUE** : Wenn **FALSE**

Typische Probleme

Diese Schreibweise ist eigentlich ein Operator, der so genannte trinare Operator. Wichtig ist, daran zu denken, dass der Operator Ausdrücke erwartet, die etwas zurück geben. Operatoren benötigen zum Arbeiten Operanden. Der folgende Ausdruck funktioniert nicht:

```
$a == 66 ? echo "A ist 66" : echo "A ist nicht 66";
```

Woran liegt das? Der Befehl echo gibt nichts zurück, er ist keine Funktion. In diesem speziellen Fall müssen Sie print verwenden. print ist eine Funktion und gibt das, was zum Browser ausgegeben wurde, auch zurück. Das wird zwar im weiteren Verlauf der Abarbeitung ignoriert, aber die Operatoren bekommen ihre Operanden, um der Semantik zu genügen. Effektiver ist natürlich das folgende Beispiel:

Listing 3.59:
shortifop:
Bedingungsoperator
mit echo anwenden

```
<?php
$test = 3;
echo $test == 3 ? 'Drei' : 'Nicht drei';
?>
```

Mehrfachauswertungen mit switch

switch()
case:
break;

Wenn Sie wie in Listing 3.55 mehrere aufeinanderfolgende Bedingungen gegen ein und dieselbe Variable testen möchten, ist der if-Befehl sehr aufwändig. Mit switch steht ein Befehl zur Verfügung, der solche Listen eleganter aufbaut:

Listing 3.60:
statement switch:
Entscheidungsbäume
mit switch

```
<?php
$stunde = date("H");
switch($stunde)
{
    case 8:
        echo "Guten Morgen";
        break;
    case 9:
        echo "Bisschen spät heute?";
        break;
    case 10:
        echo "Jetzt gibts Ärger";
        break;
    case 11:
        echo "Lass dich krankschreiben";
        break;
    default:
```

```
        echo "Sonstwann am Tage...";  
    }  
    ?>
```

Der prinzipielle Aufbau ist aus Listing 3.60 ersichtlich. Im switch-Befehl selbst steht die zu testende Variable, in den case-Abschnitten jeweils der Testwert, der auf *Gleichheit* getestet wird.

Wichtig ist der break-Befehl, der die Arbeitsweise von switch wesentlich beeinflusst. Wenn switch eine zutreffende Bedingung findet, wird nach dem case-Befehl mit der Ausführung des Codes begonnen. Weitere case-Befehle werden nicht ausgewertet, die enthaltenen Befehle werden aber ausgeführt. Der switch-Block muss also mit break explizit verlassen werden. Einen Zwang zur Anwendung gibt es natürlich nicht, manchmal ist der Effekt ja auch beabsichtigt. Das folgende Beispiel erzeugt eine kleine Anzeige, die dies demonstriert:

```
<?php  
$status = 6;  
switch($status)  
{  
    case 7: echo "|";  
    case 6: echo "|";  
    case 5: echo "|";  
    case 4: echo "|";  
    case 3: echo "|";  
    case 2: echo "|";  
    case 1: echo "|";  
}  
?>
```

Listing 3.61:
statement_nobreak:
switch ohne break

Die Ausgabe einer definierten Anzahl von Zeichen ist natürlich mit entsprechenden Funktionen oder Schleifen einfacher und eleganter lösbar. Hier ging es lediglich um eine Demonstration. Dennoch kann der gezielte Einsatz (oder Nicht-Einsatz) von break raffinierte Konstruktionen schaffen.

In Listing 3.60 wurde bereits der Befehlsbestandteil default genutzt. Dieser Zweig des switch-Befehls wird ausgeführt, wenn *keine* andere Bedingung zutrifft. Allerdings wird default erreicht, wenn hinter der zutreffenden Bedingung kein break erfolgt.

default:

Die Beschränkung des Bedingungstests auf Gleichheit mag als ernsthafte Behinderung erscheinen. Glücklicherweise kennt PHP eine erweiterte Notation der Syntax, die das Problem zumindest teilweise lindert. Dabei können mehrere Vergleiche hintereinander ausgeführt werden, die Operanden sind quasi Oder-Verknüpft:

Erweiterte Syntax:
case ... :

```
<?php  
$tag = date("w");  
switch ($tag) {  
    case 1: case 2:
```

Listing 3.62:
statement_case:
*Erweiterte Syntax
für case*

```
        echo "Wochenanfang";
        break;
    case 7: case 0: case "six":
        echo "Wochenende";
        break;
    default:
        echo "In der Woche";
}
?>
```

Trickreicher ist die Vertauschung der Operatoren, was dem switch-Befehl eine unbegrenzte Funktionalität verleiht. Das folgende Beispiel zeigt die Anwendung, die auch im Projekt *phpTemple* intensiv verwendet wird.

Listing 3.63:
switch raffiniert
erweitern: switchtrue

```
<?php
$hour = date("H");
switch (TRUE) {
    case ($hour > 6 and $hour <= 10):
        echo "Vormittags";
        break;
    case ($hour > 10 and $hour <= 14):
        echo "Mittags";
        break;
    case ($hour > 14 and $hour <= 18):
        echo "Nachmittags";
        break;
    case ($hour > 18 and $hour <= 22):
        echo "Abends";
        break;
    default:
        echo "Nachts";
}
?>
```

Das Skript zeigt den entsprechenden Tagesabschnitt in Abhängigkeit von der aktuellen Zeit an. Die Bedingungen können beliebig gestaltet werden. Es kommt hier nur darauf an, dass eine der Bedingungen TRUE werden kann, damit der Vergleich erfüllt wird. Ebenso kann natürlich auch FALSE angegeben werden, wenn die Nichterfüllung einer Bedingung interessant ist.



Hinweis

switch ist allgemein etwas schneller als Kombinationen aus if und elseif. Abgesehen von diesem Nebeneffekt sind solche Verzweigungsstrukturen auch leichter lesbar und durch die Schreibweise weniger anfällig gegen Tippfehler.

3.3.3 Schleifen

Schleifen benötigen Sie, um Programmteile mehrfach durchlaufen zu lassen. Neben der Einsparung an Tipparbeit ist vor allem die variable

Festlegung der Schleifendurchläufe interessant. Schleifen ohne feste Laufvariable werden durch eine Bedingung gesteuert. Der Zustand des logischen Ausdrucks bestimmt, ob die Schleife weiter durchlaufen wird oder nicht.

Schleifen mit while und do...while

Die häufigste Schleifenart ist die `while`-Schleife, die in fast jeder Programmiersprache zu finden ist. Die Bedingung wird mit jedem Eintritt in die Schleife getestet. Solange der Ausdruck `TRUE` zurückgibt, wird die Schleife durchlaufen. Wenn der Ausdruck also schon beim Eintritt in die Schleife `FALSE` ergibt, wird die Schleife überhaupt nicht durchlaufen.

while

```
<?php
$counter = 0;
$test = 6;
while ($test > $counter) {
    echo "Aktueller Zähler: $counter<br>";
    $counter++;
}
?>
```

*Listing 3.64:
statement while:
Einfachste Form
einer while-Schleife*

Die Schleife im Beispiel bricht ab, wenn `$counter` größer als 6 ist, also nach sieben Durchläufen. Beachten Sie auch, dass die Schleife überhaupt nicht durchlaufen wird, wenn die Bedingung gleich beim ersten Test nicht Wahr ist. Sie sollten es vermeiden, Variablen in Schleifen zu initialisieren und global gültige Zuweisungen vorzunehmen; dies ist schlechter Programmierstil. Zum einen ist es sinnlos, Zuweisungen mehrfach vorzunehmen, wenn der Wert sich nicht ändert. Zum anderen versagen solche Konstrukte, wenn die Bedingung anfangs nicht erfüllt ist. Es ist also weniger eine stilistische Frage, als mehr der Schutz vor potenziellen Fehlerquellen.

Achten Sie unbedingt darauf, dass einfache Schleifen eine klare Abbruchbedingung haben. Endlosschleifen können das System zum Stillstand bringen. Sie können den Prozess zwar immer durch `kill` beenden (Endlosschleifen bringen Linux oder Windows NT/2000 nicht zum Absturz), aber bei einem remoten Zugriff kann sich das zu einer zeitraubenden Prozedur entwickeln.



Neben der Anfangsbedingung ist derselbe Ausdruck aber auch für die Abbruchbedingung zuständig. Auch hier gilt es, bestimmte Regeln zu beachten.

Die Problematik der Abbruchbedingung kann umgangen werden, indem zusätzlich ein Notausstieg eingebaut wird. Das folgende Beispiel zeigt eine fehlerhaft programmierte Schleife – die Abbruchbedingung wird regulär nie erfüllt. Der Notausstieg verwendet den schon bekannten `break`-Befehl, der die Ausführung an die nächsthö-

re Programmebene zurückgibt; dies ist normalerweise der Befehl, der auf die schließende Klammer folgt.

*Listing 3.65:
statement break:
Notausstieg aus einer
while-Schleife mit
Hilfe einer if-
Bedingung und
break.*

```
<?php
$counter = 10;
$test = 6;
while ($counter > $test) {
    echo "Aktueller Zähler: $counter<br>";
    $counter++;
    if ($counter == 50) break;
}
?>
```

Der Begriff Notausstieg sollte hier nicht überbewertet werden. Die Anwendung des break-Befehls ist eine reguläre Programmieretechnik. Als Anfänger sollten Sie sich aber vielleicht den einen oder anderen break-Befehl einbauen, um sicher durch alle Schleifen zu kommen. Profis werden besser einschätzen können, ob die gewählten Schleifenbedingungen allen Programmsituationen genügen werden.

do ... while

Der Test der Bedingung am Anfang hat einen wesentlichen Nachteil, wenn der Inhalt des Blocks für die weitere Programmfortsetzung unbedingt erforderlich ist. Es ist möglich, dass die Bedingung so wirkt, dass der Inhalt nie durchlaufen wird. Um das sicherzustellen, gibt es die Konstruktion `do { block } while (Bedingung)`. Der einzige Unterschied ist die Reihenfolge der Abarbeitung. Zuerst wird die Schleife einmal durchlaufen und am Ende die Abbruchbedingung getestet. Auch dann, wenn die Abbruchbedingung beim Schleifeneintritt FALSE ist, wird der Block mindestens einmal ausgeführt.

*Listing 3.66:
statement dowhile:
do...while-Schleife*

```
<?php
$counter = 0;
$test = 6;
do {
    echo "Aktueller Zähler: $counter<br>";
    $counter++;
} while ($counter <= $test)
?>
```

Der Einsatz von break ist natürlich genauso und aus demselben Grund möglich.

continue

Wenn Schleifen aus dem Block heraus beendet werden können, wäre auch ein forciertes Auslösen des Bedingungstests sinnvoll. Dies können Sie mit dem Befehl `continue` erreichen. Der Befehl setzt die Ausführung sofort mit dem Test der Schleifenbedingung fort. Ist die Schleifenbedingung in diesem Augenblick FALSE, wird die Schleife mit dem Befehl `continue` verlassen.

```
<?php
$counter = 0;
$test = 10;
while ($counter < $test) {
    if ($counter % 2) {
        $counter++;
        continue;
    }
    echo "Aktueller Zähler: $counter<br>";
    $counter++;
}
?>
```

Listing 3.67:
statement continue:
continue anwenden

Das Beispiel in Listing 3.67 zeigt eine Anwendung, in der mit Hilfe einer weiteren if-Bedingung ein Teil des Schleifenblocks ausgeblendet wird. Die Schleife zeigt hier nur gerade Zahlen an. Es ist durchaus möglich und oft sinnvoll, break und continue gemeinsam zu verwenden.

Alternative Syntax

Auch für die while-Schleifen existiert die alternative Syntax, die Sie schon beim if-Befehl kennen gelernt haben. Die Ausgabe von Text zum Browser erfolgt häufig in Schleifen. Sie können die folgenden Varianten anwenden:

while :
endwhile

```
<?php
$counter = 0;
$test = 10;
while ($counter < $test):
    ?>
    Aktueller Zähler: <? echo $counter ?><br>
    <? $counter++; ?>
<? endwhile ?>
```

Listing 3.68:
statement endwhile:
Alternative Syntax
mit endwhile

Für do...while können Sie diese Syntax nicht benutzen. Hier existiert ohnehin ein schließender Befehl, die Einsparung der Klammern zur Blockbildung ist nicht vorgesehen. Abgesehen davon ist die alternative Form nicht konform mit der gesamten übrigen Syntax und kann bei künftigen Versionen durchaus entfallen. Fremde Leser von Skripten sind vielleicht irritiert. Bleibt als Schlussfolgerung: Gut zu wissen, dass es das gibt, aber in Praxis möglichst vermeiden – was im übrigen auch für die alternative Syntax von if und for gilt.



Hinweis

Zählschleifen mit for

Die vorangegangenen Beispiele dienten vor allem der Erläuterung der Syntax der Befehle, die feste Vorgabe von unteren und oberen Grenzen ist keine typische Anwendung der while-Schleifen. In solchen Fällen setzen Sie besser for-Schleifen ein. Die Abbruchbedingung ist

allerdings auch hier ein normaler logischer Ausdruck. Zusätzlich wird eine numerische Variable mitgeführt – die Zählvariable.

Als Variablennamen kommen typischerweise einzelne Buchstaben wie *\$j*, *\$k* oder *\$i* zum Einsatz. Dies ist keine Vorschrift oder gar Eigenheit von PHP, sondern dient lesbaren Skripten.

for-Schleife

Die for-Schleife ist komplexer als die bisher behandelten Varianten:

Syntax

```
for(start, bedingung, iteration);
```

Dies ist die einfachste Form der Anwendung. Die folgende Schleife zählt einfach von 0 bis 10:

*Listing 3.69:
statement_for:
Einfachste for-
Schleife*

```
<?php
for($i=0; $i <= 10; $i++) {
    echo "i ist jetzt: $i<br>";
}
?>
```

Die Variable in den drei Elementen der for-Schleife muss nicht durchgehend verwendet werden. Dies ist zwar im Hinblick auf lesbare Skripte zu empfehlen, notwendig ist es jedoch nicht, wie das folgende Beispiel zeigt:

*Listing 3.70:
statement_for2:
Raffiniertere for-
Schleife: Abbruch-
bedingung und Ite-
ration nutzen unter-
schiedliche Variablen*

```
<?php
$j = 0;
for($i=0; $i <= 10; $j++) {
    echo "$i. j ist jetzt: $j<br>";
    $i += 2;
}
?>
```

**for(;;)
continue
break**

Alle drei Parameter der for-Schleife sind optional. Ohne Iterationsvariable wird die Schleife endlos durchlaufen. Sie können in diesem Fall wieder auf **break** zurückgreifen, um die Schleife mit einer zusätzlichen Bedingung zu verlassen. Ebenso kann der Test der Abbruchbedingung und die Ausführung des Iterationsschrittes mit **continue** erzwungen werden.

*Listing 3.71:
statement_for3.php:
Varianten der for-
Schleife*

```
<?php
for(;;) {
    $i++;
    if ($i % 2) {
        continue;
    }
    echo "i ist jetzt: $i<br>";
    if ($i > 5) break;
}
?>
```

Der flexible Umgang mit den Schleifenvariablen kennt praktisch keine Grenzen. Auch das folgende Beispiel ist syntaktisch korrekt. Eingesetzt werden hier gleich zwei Variablen:

```
<?php
for($i=0,$j=10;$i<$j;$i++) {
    $j--;
    echo "i ist jetzt: $i<br>";
}
?>
```

*Listing 3.72:
statement_for4: for-Schleife mit mehreren Variablen*

Es spielt offensichtlich keine Rolle, ob hier Variablen zum Einsatz kommen oder nicht. Wenn Sie nur einfach eine Liste ausgeben möchten, kann der entsprechende Befehl sehr knapp ausfallen:

```
<?php
for($i=0, $j=10; $i<$j; $i++, $j--, print "$i<br>");
?>
```

*Listing 3.73:
statement_for5: All-In-One-Konstrukt*

```
1
2
3
4
5
```

*Abbildung 3.15:
Abbruch in der Mitte: Schleifen mit zwei Variablen*

Innerhalb des for-Befehls können also weitere Befehle, durch Komma getrennt, eingebaut werden. Im Beispiel wird übrigens nicht zufällig der Befehl print anstatt des flexibleren echo verwendet. Wie bei der Befehlsbeschreibung bereits erläutert, gibt echo nichts zurück, während print die Ausführung im Erfolgsfall mit TRUE quittiert. Normalerweise wird dieser Rückgabewert weggeworfen. Die for-Schleife erwartet jedoch von jedem direkt eingegebenen Wert, dass er sich als Ausdruck verhält – Ausdrücke geben immer etwas zurück. An dieser Stelle kann echo also nicht funktionieren. Versuchen Sie es dennoch, erhalten Sie einen Laufzeitfehler.

Alternative Syntax der for-Schleife

Auch innerhalb von for-Schleifen werden Ausgaben per HTML erfolgen. In solchen Fällen können Sie sich wieder viel Tipparbeit ersparen, indem die alternative Syntax mit endfor genutzt wird:

```
<? for($i=0; $i <= 10; $i++): ?>
i ist jetzt: <? echo $i ?><br>
<? endfor; ?>
```

**for():
endfor**

Alle vorher bereits dargestellten Konstruktionen sind natürlich mit dieser Syntax kombinierbar.

Arrays mit foreach bearbeiten

foreach für Arrays

Ab PHP 4 steht eine neue Form der for-Schleife zur Verfügung: foreach. Damit wird ein Array durchlaufen. Gegenüber der while-Schleife mit list und each vereinfacht sich die Syntax deutlich:

Syntax

```
foreach( array as element ) {  
    ...  
}
```

Dabei wird das Array *array* von Anfang bis Ende durchlaufen und bei jedem Schleifendurchlauf wird das aktuelle Element der Variablen *arrayelement* zugewiesen. Jedes Arrayelement kann wiederum ein Array sein, dann könnte mit einer verschachtelten foreach-Schleife auch dieses Array ausgewertet werden.

Zuerst ein einfaches Beispiel mit einem assoziativen Array:

Listing 3.74:
*foreach*echo: Array
ausgeben

```
<?php  
$myarray = array("x1","x2","x3","x4");  
foreach($myarray as $xwert) {  
    echo "$xwert <BR>";  
}  
?>
```

Typischerweise ist es erlaubt, einzelne Elemente eines solchen Arrays mit unterschiedlichen Datentypen zu belegen. Das können auch weitere Arrays sein. Das folgende Beispiel zeigt, wie solche komplexen Gebilde auf einfachste Weise zur Anzeige gebracht werden können:

Listing 3.75:
statement foreach:
Auslesen komplexer
Arrays mit foreach

```
<?php  
$myarray = array("x1",45,array(2,4,6,8),"x4");  
foreach($myarray as $val)  
{  
    if (gettype($val) == "array")  
    {  
        echo "Inneres Array: <BR>";  
        foreach($val as $innerval)  
        {  
            echo " -- $innerval<BR>";  
        }  
        echo "Ende inneres Array.<P>";  
    } else {  
        echo "$val<BR>";  
    }  
}  
?>
```

Der Ausgabe mangelt es zwar an der Formatierung, die Informationen über die Struktur des Arrays sind aber klar erkennbar.

```
x1
45
Inneres Array:
-- 2
-- 4
-- 6
-- 8
Ende inneres Array.

x4
```

Abbildung 3.16:
Arraystruktur
ausgeben

Assoziative Arrays verarbeiten

Wenn Sie Arrays mit Schlüssel/-Wertepaaren bauen, kann foreach mit einer erweiterten Syntax diese Paare direkt auslesen. Die grundlegende Syntax lautet:

foreach (. as . => .)

```
foreach(array as schlüssel => wert) {
    ...
}
```

Syntax

Hier wird der Operator => eingesetzt, der schon bei der Konstruktion des Arrays verwendet wurde.

```
<?php
$array = array("Name" => "Krause",
               "Ort" => "Berlin",
               "PLZ" => 12683,
               "Vorwahl" => "030");
foreach($array as $key => $val)
    echo "Feld &lt;$key&gt; hat den Wert <b>$val</b><br>";
?>
```

Listing 3.76:
statement foreach2:
Assoziative Arrays
mit Schlüssel-
/Wertpaaren mit
foreach auslesen

Mehr Information zu Arrays und Anwendungsbeispiele für foreach finden Sie im ➞ Abschnitt *Erweiterte Array-Funktionen* ab Seite 185.

3.3.4 Benutzerdefinierte Funktionen

Funktionen sind kleine Programmabschnitte oder Module, die PHP-Code enthalten. Sie werden, ebenso wie die bereits erläuterten eingebauten Funktionen, aus anderen Funktionen oder der »Stammebene« des Skripts heraus aufgerufen. Funktionen können Werte zurückgeben, beispielsweise das Ergebnis einer Berechnung. Werte sind alles, was Sie in Variablen speichern können, also auch komplexe Gebilde wie Arrays oder Objekte.

Im Zusammenhang mit PHP werden Sie vielleicht Prozeduren vermissen – vor allem Umsteiger von VBScript sind davon betroffen. Prozeduren sind in PHP nicht eigenständig als Schlüsselwort zu vereinbaren. Wenn sich die Notwendigkeit einer Prozedur ergibt, wird eine Funktion geschrieben und der Rückgabewert ignoriert bzw. das für die Rückgabe zuständige Schlüsselwort return weggelassen.



Hinweis

Funktionsdefinition

function()

Nutzerdefinierte Funktionen werden durch das Schlüsselwort `function` eingeleitet. Die Syntax kann folgendermaßen aussehen:

Syntax

```
function meinefunction($param, $param2, ...)
{
    // Hier werden Befehle ausgeführt
    return $rueckgabe;
}
```

Der Funktionskopf besteht aus dem Namen der Funktion (hier: *meinefunction*) und in runden Klammern einer Auflistung der erwarteten Parameter. Diese Werte werden der Funktion beim Aufruf übergeben. Ein Zwang zur Übergabe von Werten besteht nicht, schreiben Sie einfach ein leeres Klammerspaar.

return()

Der Befehl `return` enthält als Parameter den Rückgabewert. Dies kann ein Ausdruck oder eine einzelne Variable sein. An welcher Stelle innerhalb der Funktion Sie `return` einsetzen, spielt keine Rolle. Auch die mehrfache Notation ist zulässig – hier wird nach dem Motto »Wer zuerst kommt, siegt« verfahren und die Funktion wird sofort verlassen. Aus Gründen sauberer Programmierung sollten Sie aber `return` trotzdem nur einmal und nur am Ende einsetzen.



In PHP 3 müssen Funktionen vor der ersten Verwendung im Skript geschrieben werden. Der PHP-Interpreter arbeitet das Skript von oben nach unten durch. Findet er einen Funktionsaufruf und die Definition der Funktion steht dahinter, passiert nichts.

Späte Bindung in PHP 4

In PHP 4 werden die Funktionen erst nach dem Abarbeiten des Skripts im Parser gebunden¹⁰. Dadurch spielt es keine Rolle, wann sie definiert werden.

Funktionsargumente

Die Übergabe von Argumenten an die Funktion (die dann dort Parametern entsprechen) kann auf vielfältige Art und Weise erfolgen. Im einfachsten Fall geben Sie Variablennamen an:

```
<?php
function print_value($name, $ort)
{
    echo "$name wohnt in $ort.";
}
?>
```

¹⁰ Dies ist ein Fachausdruck, der aussagt, wann die Adresse der übersetzten Funktion im Speicher ermittelt und zum Aufruf bereitgestellt wird.

Der Aufruf der Funktion kann nun erfolgen, indem Werte eingesetzt werden:

```
print_value("Max Müller", "Berlin");
```

Der Rückgabewert interessiert hier nicht, also wird auch kein return benötigt. Beim Aufruf können natürlich auch Variablen eingesetzt werden. Das folgende Beispiel ist äquivalent zu dem vorherigen:

```
$name = "Max Müller";  
$ort = "Berlin";  
print_value($name, $ort);
```

Die Variablennamen können gleich sein, da Variablen innerhalb einer Funktion lokal sind und nicht im Konflikt mit den globalen Variablen stehen.

Lokale und globale Variablen

```
<?php  
function print_value($name, $ort) {  
    echo "$name wohnt in $ort.<br>";  
    $name = "Neuer Name";  
    $ort = "Neuer Ort";  
}  
$name = "Max Müller";  
$ort = "Berlin";  
print_value($name, $ort);  
print_value($name, $ort);  
?>
```

Listing 3.77:
function scope:
Verhalten lokaler und globaler Variablen beim Funktionsaufruf

Die Veränderung der übergebenen Variablen kann aber manchmal erwünscht sein. So könnte eine Funktion sämtliche übergebenen Werte in HTML fett schreiben:

Übergabe per Referenz

```
<?php  
function make_b(&$html) {  
    $html = "<b>".$html."</b>";  
}  
$ausgabe = "Test";  
echo "$ausgabe<br>";  
make_b($ausgabe);  
echo $ausgabe;  
?>
```

Listing 3.78:
function byref:
Anwendung einer Funktion mit referenzierten Argumenten

Der Name der Variablen spielt keine Rolle. Entscheidend ist die Kennzeichnung der Argumente mit &. In diesem Fall wird der globalen Variablen der neue Wert zugewiesen. Diese Vorgehensweise wird als Parameterübergabe durch Referenz bezeichnet (*ByRef*). Der normale Weg ist die Übergabe von Werten (*ByVal*), Änderungen wirken sich dann nicht auf die Quelle aus.

Diese Technik kann auch als Trick verwendet werden, um mehrere Werte zurückzugeben. Eine Funktion kann nur einen Wert zurückgeben. Wenn Sie aber mehrere referenzierte Argumente haben, können

Mehrere Argumente ändern

Sie jeden Wert direkt ändern und so außerhalb der Funktion wirksam werden lassen.

Als Wert wird aber auch ein Array akzeptiert. Der folgende Abschnitt zeigt, wie Sie mehrere Werte mit Hilfe von Arrays zurückgeben und wie Sie Argumente übergeben, deren Anzahl Sie zum Entwurfszeitpunkt nicht kennen.

Probleme bei der Argumente-übergabe

Das Beispiel in Listing 3.78 mag Ihnen zu umständlich erscheinen. Die folgende Zeile ist kompakter, verständlicher und falsch:

```
echo make_b($ausgabe);
```

Syntaktisch ist dies richtig, der PHP-Parser wird nicht mit einer Fehlermeldung reagieren. Das Problem ist das Verständnis für die Parameterübergabe. Die Beispielfunktion *make_b* gibt nichts zurück – sie ändert nur eine übergebene Variable. Der Befehl *echo* erwartet aber Werte, die er an den Browser senden kann. An die eingeschlossene Variable *\$ausgabe* kommt er nicht heran. Diese wird zwar geändert, das Ergebnis aber quasi verworfen. Wenn Sie die Anwendung dennoch in dieser Form benötigen, setzen Sie *return* ein:

```
function make_b($html) {
    return "<b>".$html."</b>";
}
```

Wenn eine Funktion die Übergabe durch Referenz nicht unterstützt (weil sie beispielsweise in einer Bibliothek versteckt ist), können Sie dieses Verhalten dennoch erzwingen. Stellen Sie das *&*-Zeichen einfach in den Funktionsaufruf. Das bereits gezeigte Beispiel aus Listing 3.78 funktioniert also auch so:

Listing 3.79:
funcrefbycall:
Referenzen beim
Aufruf erzeugen

```
<?php
function make_b($html) {
    $html = "<b>".$html."</b>";
}
$ausgabe = "Test";
echo "$ausgabe<br>";
make_b(&$ausgabe);
echo $ausgabe;
?>
```

Optionale Parameter

Optionale Parameter kamen in PHP 3 zum Einsatz, um variable Argumentelisten zu imitieren. Sie können generell einzelne Argumente beim Aufruf weglassen. In diesem Fall wird die betreffende Variable nicht initialisiert. Um zusätzliche Tests zu vermeiden und Laufzeitfehler innerhalb der Funktion nicht zu provozieren, können Sie jedes Argument mit einem Standardwert belegen. Dazu wird der Funktionskopf um eine Zuweisung erweitert:

```
function print_value($name = "Name", $ort = "Ort")
```

Eleganter ist die Verwendung von Konstanten:

```
define("NONAME", "Kein Name");
...
function print_value($name = NONAME, $ort = "Ort")
```

Anfängern bereiten optionale Parameter oft Schwierigkeiten. Der Umgang damit ist einfach, solange die Anwendung in der gezeigten Form erfolgt. Es ist jedoch auch möglich, einige Parameter mit Standardwerten zu belegen und andere nicht. Das folgende Beispiel funktioniert problemlos:

```
function print_value($name, $ort = "Ort")
```

Der umgekehrte Fall erscheint ebenso einfach, führt jedoch zu einem Laufzeitfehler:

```
function print_value($name = "Name", $ort)
```

Dieses Beispiel funktioniert nur, wenn tatsächlich beide Parameter übergeben werden. Der folgende Aufruf kann nicht funktionieren:

```
print_value("Berlin");
```

Woher soll PHP wissen, dass Sie den zweiten Parameter meinen? Fazit: optionale Parameter müssen im Funktionskopf immer rechts stehen und von rechts auflösbar sein.

Variable Argumentelisten

In PHP 3 konnten noch keine variablen Argumentlisten verwendet werden. Um diesen Nachteil zu umgehen, wurden neben den bereits beschriebenen optionalen Parametern Arrays eingesetzt. Sie können die Anzahl der Argumente dann mit Hilfe von Array-Funktionen bestimmen. Diese Technik ist auch unter PHP 4 noch aktuell, denn manchmal ist es einfacher als der direkte Umgang mit variablen Listen.

Das folgende Beispiel zeigt eine Funktion, die eine variable Anzahl Elemente übernimmt und formatiert zurückgibt:

```
<?php
function many_html($tag, $argv) {
    $argc = count($argv);
    for ($i = 0; $i < $argc; $i++) {
        $result .= "<".$tag.">".$argv[$i]."</".$tag."> ";
    }
    return $result;
}
$wert = array("Dies ", "ist ", "ein ", "test ");
echo "Fett: " . many_html("b", $wert);
echo "<br>";
echo "Kursiv: ";
echo many_html("i", $wert);
```

**Probleme mit
optionalen
Parametern**



Listing 3.80:
function varlist:
Variable Argument-
listen mit Arrays

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

```
echo "<BR>";
?>
```

Der Trick besteht in der Verwendung von array. Die Anzahl der Argumente kann leicht mit der Funktion count ermittelt werden. Für die Übergabe wird das Array mit der Funktion array aus Einzelwerten erzeugt. Wie viele dies sind, spielt nun keine Rolle mehr.



In PHP 4 sind variable Argumentelisten zulässig. Dies wird bei internen Funktionen wie beispielsweise max verwendet. Die Funktion ermittelt den maximalen Wert einer beliebig langen Liste von Argumenten.

func_num_args()
func_get_arg()
func_get_args()

Die Definition ist besonders einfach: sie ist nicht nötig. Wie wollen Sie auch die Argumente nennen, wenn Sie deren Anzahl nicht kennen? Es gibt deshalb spezielle Funktionen, mit denen der Parameterblock untersucht werden kann:

- func_num_args

Diese Funktion gibt die Anzahl der Parameter zurück.

- func_get_arg

Hiermit ermitteln Sie einen konkreten Parameter, die Auswahl erfolgt durch eine Nummer.

- func_get_args

Diese Funktion gibt ein Array mit den Parametern zurück.

Das folgende Beispiel implementiert zwei Funktionen, die beliebig viele Parameter akzeptieren. Realisiert wird eine mathematische Funktion zur Berechnung der Summe und des Durchschnitts.

Listing 3.81:
function varparam:
Funktionen mit
variablen Argument-
listen

```
<?php
function sum($arr) {
    $str = join("+", $arr);
    eval("\$sum = $str;");
    return $sum;
}

function avg() {
    return sum(func_get_args()) / func_num_args();
}

echo avg(13, 12, 7, 8, 23);
?>
```

Die Funktion *sum* wird mit dem konventionellen Verfahren aus PHP 3 versorgt: mit einem Array. Das Array wird zu einer Zeichenkette zusammengefasst; als Trennzeichen wird das Pluszeichen eingesetzt. Der Befehl eval (dies ist keine Funktion) wertet eine Zeichenkette als PHP-Code und führt ihn aus. Praktisch wird damit die Summe berechnet und der Variablen *\$sum* zugewiesen, die dann als Rückgabewert ver-

wendet wird. Diese Implementierung wird hier verwendet, um `func_get_args` verwenden zu können.

12.6

Die Funktion `avg` berechnet den Durchschnittswert. Dazu wird die mit `sum` ermittelte Summe durch die Anzahl der Werte dividiert. Die Anzahl ermittelt `func_num_args`.

Im Beispiel wurde gezeigt, dass keine Parameterangaben im Kopf der Funktionsdeklaration notwendig sind. Wenn Sie dennoch Parameter angeben, werden diesen Argumente vom Anfang der Liste zugewiesen. Unabhängig davon werden alle Werte von den Parameterfunktionen verwaltet. Das folgende Beispiel gibt »14« zurück:

```
<?php
function num($first) {
    echo $first . func_get_arg(3);
}
num(1, 2, 3, 4, 5);
?>
```

Abbildung 3.17:
Ausgabe des Skripts
aus Listing 3.81

Listing 3.82:
Zusätzliche Aus-
wertung des ersten
Parameters:
[function varparam2](#)

Rückgabe mehrerer Werte

Die Rückgabe mehrerer Werte ist nicht vorgesehen. Da es aber sonst keine Beschränkung des Typs gibt – sowohl Arrays als auch Objekte sind erlaubt –, bietet sich ein einfacher Trick zur Übergabe mehrerer Werte an.

Eine Variante kennen Sie bereits: die Übergabe von Variablen als Referenz mit vorangestelltem `&`-Zeichen. Daneben können Sie auch als Rückgabewert ein Array einsetzen:

```
<?php
function make_b(&$html) {
    $html = array("<b>", $html, "</b>");
}
make_b($html);
list($opentag, $html, $closetag) = $html;
echo "$opentag$html$closetag";
?>
```

Listing 3.83:
Rückgabe mehrerer
Werte mit Arrays:
[function array](#)

Die Auflösung wird mit `list` vorgenommen. Oft ist diese Schreibweise offensichtlicher und leichter zu verstehen als mit referenzierten Parametern. Programmierern mit Erfahrung in höheren Sprachen, wie C, mag dies dagegen unheimlich erscheinen.

Rekursion

PHP erlaubt rekursive Funktionsaufrufe. Dabei ruft eine Funktion sich selbst auf. Das wird beispielsweise benötigt, um verschachtelte Struk-

turen auszugeben. Dabei kann es sich um Verzeichnisse oder Menüführungen handeln. Wenn Sie viel mit Arrays experimentieren, mag Ihnen das folgende Beispiel gefallen. Es stellt Arrays in Arrays dar und der Funktion ist es egal, wie viele Verschachtelungsebenen Sie verwenden.

*Listing 3.84:
function_recursion:
Anwendungsbeispiel
für eine rekursive
Funktion zur
Ausgabe von beliebig
verschachtelten
Arrays*

```
<?php
function array_show($arr) {
    static $shift = 1;
    foreach($arr as $key => $val) {
        if (is_array($key)) {
            $shift++;
            array_show($key);
        } else {
            echo str_repeat("--&nbsp;", $shift);
            echo "<b>$key</b>";
            echo "<br>";
        }
        if (is_array($val)) {
            $shift++;
            array_show($val);
        } else {
            echo str_repeat("==&nbsp;", $shift) . $val;
            echo "<br>";
        }
    }
    $shift > 1 ? $shift-- : $shift = 1;
}
$arrCheck = array("key1" => "val1",
                  "key2" => "val2",
                  array(1,2,3),
                  "keyX" => array(33 => 66,
                                  66 => 99),
                  4,
                  array("M" => "m",
                        "N" => "n",
                        array("Z1", array("Z2" => "Z3"), "Z4"),
                        )
                  );
echo "<pre>";
array_show($arrCheck);
echo "</pre>";
?>
```

Die Funktion ist zusätzlich mit der Anzeige der Verschachtelungstiefe durch Einrückung versehen. Das Array *\$arrCheck* dient nur der Demonstration. Damit die Einrückung bei jedem Aufruf der Funktion erhalten bleibt, wird die Variable *\$shift*, in der die Tiefe gespeichert wird, statisch gesetzt (static). Die Funktion ist im Kern auf drei Zeilen beschränkt:

```
if (is_array($key)) {
    $shift++;
    array_show($key);
}
```

Zuerst wird geprüft, ob der von foreach gelieferte Wert noch ein Array ist – das ist dann ein verschachteltes Array. Ist das der Fall, wird die Tiefe erhöht und die Funktion ruft sich mit dem Teilarray selbst auf. Wurden keine Arrays mehr gefunden, gelangt die Funktion zum Ende, verringert dort die Tiefe und kehrt zum aufrufenden Punkt zurück:

```
$shift--;
```

Die nebenstehende Abbildung zeigt, wie das Ergebnis mit den Musterdaten aussieht. Schlüssel sind fett, Werte normal und mit --Zeichen dargestellt. Beachten Sie, dass assoziative Arrays fehlende Schlüssel durch numerische Indizes auffüllen (beispielsweise die 1 vor der alinstehenden 4).

```
-- key1
== val1
-- key2
== val2
-- 0
-- 0
== 1
-- 1
== 2
-- 2
== 3
-- keyX
-- 33
== 66
-- 66
== 99
-- 1
== 4
-- 2
-- M
== m
-- N
== n
-- 0
-- 0
== 21
-- 1
-- 22
== 23
-- 2
== 24
```

Erweiterte Syntax

Auch wenn sich der Sinn nicht sofort erschließt, sei an dieser Stelle eine andere Form des Funktionsaufrufes dargestellt. Es ist möglich, Funktionen über eine Referenz aufzurufen, das heißt, der Inhalt einer Variablen bestimmt den Funktionsnamen. Das folgende Beispiel soll dies verdeutlichen:



```
<?php
function firstfunction() {
    echo "Eine spannende Funktion... <br>";
    return TRUE;
}
function secondfunction() {
    echo "Eine zweite spannende Funktion... ";
    return TRUE;
}

$name = "first";
$name = "second";
$type = "function";
$dynfunction = $name.$type;
$ok = $dynfunction();
$secondfunction = $name.$type;
```

Listing 3.85:
func_dynname:
Funktionsnamen
dynamisch erstellen

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

```
$ok = ${'secondfunction'}();  
?>
```

Die Syntax des Aufrufes ist einfach, kann aber in manchem Kontext gut zur Klammerwüste ausarten. Deshalb nochmal in vereinfachter Form:

```
$funktionsname = "myfunction";  
$funktionsname(Parameter);
```

Dies entspricht dem folgenden Aufruf:

```
myfunction(Parameter);
```

Einsatz

Solche Konstruktionen werden oft für sogenannte Wrapper verwendet. Dies sind Programme, die einen universellen Funktionsaufruf einer von Bedingungen abhängigen Auswahl spezieller Funktionen zuweisen. Manche Prüfprogramme für Kreditkartennummern arbeiten damit. Nach dem eine Funktion den Typ der Karte erkannt hat, erfolgt eine Verzweigung auf einen der vielen möglichen Algorithmen. Statt umfangreicher switch-Bäume bleibt es bei einem Aufruf mit variablem Namen.

3.4 Objektorientierte Programmierung

Objektorientiert programmieren setzt voraus, das die dahinter liegenden Paradigmen verstanden wurden. Einführungen in diese Welt leiden meist unter einer Flut neuer Begriffe, die sich gegenseitig erklären – wenig hilfreich, wenn ein Ansatzpunkt fehlt.

3.4.1 Einführung



Ein Blick in die reale Welt hilft vielleicht, das Konzept besser aufzunehmen. In der realen Welt ist alles ein Objekt. Jedes Ding besteht aus einem Zustand, beschrieben durch Eigenschaften und zulässigen Verhaltensregeln. Dabei können Gegenstände sehr simpel sein, komplexere Objekte wie Tiere oder Menschen dagegen kaum noch erfassbar.

In der Programmierung werden solche zusammen aufgeschriebenen Gebilde aus Eigenschaften und Ablaufregeln als Objekte bezeichnet. In klassischen Programmen speichern Sie Daten in Variablen und geben dann durch Programmanweisungen und Funktionen Regeln vor, nach denen das Programm abläuft. Interaktion mit Schnittstellen schafft dann die Vielfalt der Abläufe. Objektorientierte Programme bestehen aus Sammlungen von Objekten.

Was sind Klassen?

Die Idee dahinter ist die Zusammenfassung gleichartiger oder ähnlicher Strukturen und zusammengehörender Elemente. Solche Gebilde werden als Klassen bezeichnet. In PHP werden sie mit dem Schlüsselwort `class` definiert. Es werden also nicht immer wieder Objekte definiert, wenn man sie benötigt, sondern nur ein Mal – als Klasse. Wenn der Programmierer dann ein Objekt benötigt, beispielsweise um eine Programmfunktion auszulösen, leitet er aus der Klasse ein Objekt ab und verwendet dies. Dieser Vorgang wird als »instanziiieren« bezeichnet. Das Objekt ist eine »Instanz« der Klasse. Aus einer Klasse kann man beliebig viele Objekte ableiten. Diese sind voneinander völlig getrennt und werden für sich wiederum in Variablen gespeichert. Objekte werden von den Klassen mit `new` abgeleitet.

Vererbung

Wenn Sie über objektorientierte Programmierung lesen, werden Sie sehr bald auf das Wort Vererbung stoßen. Bislang spielte es hier keine Rolle. Sie müssen Vererbung nicht verwenden. Erst bei größeren Projekten wird es sinnvoll sein. Was aber hat es damit auf sich?

Wenn Sie eine Klasse definiert haben, kommt vielleicht der Zeitpunkt, wo eine Erweiterung notwendig wird. Vielleicht fallen Ihnen aber auch mehrere Variationen dieser Klasse ein. Anstatt nun mehrere Klassen zu definieren, ist es besser, eine Basisklasse zu entwerfen und daraus Subklassen abzuleiten, die nur die geänderten Eigenschaften und Methoden enthalten. Damit die Subklassen überhaupt eine Funktionalität haben, müssen Sie diese von einer Mutterklasse erben. Dieser Vorgang wird Vererbung genannt.

In PHP wird die Vererbung mit der Anweisung `extends` gesteuert. Dabei definieren Sie eine Klasse und geben mit `extends` an, von welcher Klasse zusätzliche Eigenschaften und Methode geerbt werden sollen. Aus Ketten solcher Vererbungen entsteht eine Hierarchie. Dies ist auch mit PHP möglich. Ebenso können Methoden überschrieben werden. Tritt in einer Klasse, die bereits Methoden geerbt hat, eine Methode gleichen Namens auf, überschreibt diese die ursprüngliche Methode.

PHP und Polymorphie

Wenn Sie bereits andere, höhere Sprachen kennen, haben Sie vielleicht schon den Begriff der Polymorphie gehört. Normalerweise wird für Methoden eine einheitliche Schnittstelle definiert – also Parameter und Rückgabewerte werden festgelegt. Tritt ein »Erbfall« auf, müssen sich die Methoden, die andere überschreiben, an die Schnittstellendefinition halten. Dies erzwingt einen sauberen Programmierstil. Polymorphie erlaubt dagegen explizit – durch Angabe eines Schlüsselwor-

Eigenschaften weitergeben

Vererbung steuern

Polymorphie wird nicht unterstützt

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

tes – die Veränderung der Schnittstelle in Subklassen. PHP erlaubt dies sowieso, weil Schnittstellen überhaupt nicht definiert werden. Streng genommen unterstützt PHP Polymorphie nicht, sondern verhält sich immer so, als ob dies erlaubt wäre.

3.4.2 Definition einer Klasse

class

Die objektorientierte Programmierung beginnt natürlich mit der Definition einer Klasse. In PHP verwenden Sie dazu die Anweisung `class`. Das Beispiel führt Datumsberechnungen aus. Das geht ohne Klassen zwar auch, sieht aber nicht mehr so elegant aus.

Listing 3.86:
classfirst: Eine Klasse
definieren und
verwenden

```
<?php
class printdate
{
    var $language;
    var $datetime;

    function weekday()
    {
        $old = setlocale(LC_TIME, $this->language);
        echo strftime('%A', $this->datetime);
        setlocale(LC_TIME, $old);
    }

    function month()
    {
        $old = setlocale(LC_TIME, $this->language);
        echo strftime('%B', $this->datetime);
        setlocale(LC_TIME, $old);
    }
}

$dd = new printdate;
$de = new printdate;
$dd->language = 'ge';
$de->language = 'fr';
$dd->datetime = $de->datetime = time();

$dd->weekday();
echo '<hr>';
$de->weekday();
?>
```

Wie es funktioniert

Das Skript zeigt sowohl die Klassendefinition als auch die Instanziierung der Objekte. Die Klasse *printdate* kennt zwei Eigenschaften und zwei Methoden. Sie soll Wochentage und Monatsnamen in verschiedenen Sprachen ausgeben.

Die Eigenschaften sind:

- *\$language*. Eine Codierung, welche Sprache verwendet wird.
- *\$datetime*. Das Datum, zu dem Wochentag oder Monat ausgegeben wird.

Die Methoden sind die Handlungsanleitungen:

- *weekday()*. Hiermit erfolgt eine Ausgabe des Wochentags
- *month()*. Diese Methode gibt den Monatsnamen aus.

Das Objekt besteht also aus zwei Eigenschaften und zwei Methoden. Die Klasse wird dem durch `class` gebildeten Mantel erstellt. Innerhalb der geschweiften Klammern werden die Eigenschaften mit der Anweisung `var` definiert. Eigenschaften stehen immer als Variablen zur Verfügung. Methoden werden dagegen wie Funktionen erzeugt. Parameter und Rückgabewerte können Sie ebenso verwenden, wie Sie dies von normalen Funktionen gewohnt sind.

Umgang mit den Definitionen in der Klasse: `$this`

Der Zugriff auf Eigenschaften und Methoden innerhalb der Klasse verwendet die spezielle Variable `$this`. Wenn Sie sich »virtuell« innerhalb der Klassendefinition befinden, existiert ja eigentlich noch kein Objekt. `$this` nimmt diese Instanziierung quasi voraus und verweist auf die Klasse selbst. Praktisch läuft dieser Vorgang natürlich erst dann ab, wenn ein Objekt erzeugt wurde. Wenn viele Objekte existieren, gibt es auch ebenso viele `$this`-Instanzen. Zur Trennung von Objekt und Eigenschaft oder Methode wird der Operator `->` eingesetzt. Dies gilt für `$this` und jede andere Objektvariable. Ob es sich um eine Eigenschaft oder Methode handelt, erkennt PHP alleine anhand der runden Klammern.

Achten Sie darauf, bei Methoden die runden Klammern nicht zu vergessen – auch wenn keine Parameter folgen. Wenn Sie diese weglassen, erkennt PHP eine Eigenschaft. Wie bei Variablen üblich, werden nicht vorhandene Variablen beim ersten Aufruf angelegt und bleiben leer, wenn sie nicht verwendet werden. Eine Fehlermeldung wird hier nicht erzeugt.

Im Beispiel erfolgt der Zugriff auf die Eigenschaften innerhalb der Klasse mit Hilfe von `$this`:

```
$old = setlocale(LC_TIME, $this->language);  
echo strftime('%A', $this->datetime);  
setlocale(LC_TIME, $old);
```

Das Objekt instanziiieren

Von der Klasse zum Objekt

Die bis hierher definierte Klasse ist noch völlig funktionslos. Sie nimmt nicht einmal Speicher in Anspruch – außer dem Platz für den Quelltext. Erst durch Instanziierung zur Laufzeit wird sie aktiv.

new

Dafür ist eine weitere Anweisung zuständig: `new`. Als Parameter wird der Name einer Klasse angegeben, als Ergebnis entsteht ein Objekt. Im Beispiel werden zwei davon erzeugt:

```
$dd = new printdate;  
$de = new printdate;
```

Die Objekte – mit den Eigenschaften *\$language* und *\$datetime* und den Methoden *weekday()* und *month()* – *\$dd* und *\$de* sind völlig unabhängig voneinander. Erst jetzt wird auch tatsächlich Speicher in Anspruch genommen.

Der Zugriff auf die Eigenschaften erfolgt wie bei Variablen. Anstatt der internen Klassenvariablen *\$this* – die hier nicht mehr existiert – muss nun natürlich die Objektvariable verwendet werden.

```
$dd->language = 'ge';  
$de->language = 'fr';  
$dd->datetime = $de->datetime = time();
```

Beachten Sie die Position des *\$*-Zeichens – es wiederholt sich nicht hinter dem *->*-Operator.

Methoden werden ebenso wie Funktionen verwendet, nur steht auch hier wieder die Objektvariable davor. Damit ist klar, in welchem Objekt die entsprechende Aktion ausgelöst wird:

```
$dd->weekday();  
echo '<hr>';  
$de->weekday();
```

Denken Sie an die runden Klammern, damit PHP den Aufruf als den einer Methode erkennt.

3.4.3 Erweiterte Techniken für Objekte

Zugriff auf Klassen und Vererbung

Mit der gezeigten Form der Definition einer Klasse und ihrer Eigenschaften und Methoden können Sie schon viel erreichen. Wenn Sie solche Klassen häufig verwenden, müssen Sie sich über die Benennung der Eigenschaften und Methoden weniger Gedanken machen. Die Klasse bildet einen abgeschlossenen Namensraum – Konflikte mit Definitionen gleichnamiger Funktionen im imperativen Teil des Skripts oder anderen Klassen können nicht auftreten.

Direkter Zugriff auf Klassen

Manchmal lohnt es sich, auch dann eine Klasse zu definieren, wenn `::` die Existenz eines Objekts gar nicht notwendig wäre. Dazu bietet PHP 4 den neuen Operator `»::«`.

```
<?php
class calc
{
    function cm2in($cm)
    {
        return $cm / 2.54;
    }

    function in2cm($in)
    {
        return $in * 2.54;
    }
}
echo calc::cm2in(3);
echo '<hr>';
echo calc::in2cm(4.5);
?>
```

Listing 3.87:
directclass: Klassen
ohne Instanziierung
verwenden

Das Beispiel zeigt eine Umrechnung von Zentimeter in Zoll (in steht für *inches*) und umgekehrt. Eine Zuweisung zu Eigenschaften und ein Erhalten eines Objektes allein deswegen wäre zu umständlich. Die direkte Verwendung bringt hier Vorteile. Vor allem müssen Sie sich keine Gedanken darüber machen, ob irgendwo anders eine Funktion *cm2in()* oder *in2cm()* existiert. Der Aufruf nennt sowohl die Klasse als auch die Funktion:

```
echo calc::cm2in(3);
```

Manche Probleme wären ohne diese Methode nicht lösbar. Wenn Sie zwei Klassendefinitionen haben, können Sie nicht auf Methode oder Eigenschaften der anderen Definition zugreifen: `new` ist hier nicht zulässig und `$this` verweist immer nur auf die eigene Klasse. Mit dem `::`-Operator können Sie jedoch auch innerhalb einer Klassendefinition den Namen einer anderen Klasse verwenden.

Wie es funktioniert

Nicht alles ist so machbar

3.4.4 Spezielle Funktionen

Des weiteren sollten Sie sich mit einigen Funktionen auseinandersetzen, die den Umgang mit Klassen unterstützen. Eine vollständige Übersicht über Syntax und Anwendungsbeispiele finden Sie in der Referenz.

Funktionen für Objekte

Methoden aufrufen

Der Aufruf von Methoden mit den Operatoren `->` und `::` ist manchmal nicht hinreichend flexibel. Sie können dies auch mit Hilfe einer Funktion initiieren:

*Listing 3.88:
call_user_method:
Methoden per
Funktion aufrufen*

```
<?php
class calc
{
    # Klassendefinition aus Listing 3.87
}
$calc = new calc;
$f[] = "cm2in";
$f[] = "in2cm";
echo call_user_method($f[0], $calc, 3);
echo '<br>';
echo call_user_method($f[1], $calc, 3);
?>
```

Wie es funktioniert

Hier werden die Funktionsnamen einem Array entnommen – der Aufruf selbst ändert sich dadurch natürlich nicht. Die Funktion `call_user_method` erwartet mindestens zwei Parameter: Den Namen einer Methode und einer Klasse. Der dritte und eine beliebige Anzahl weiterer Parameter wird an die Methode beim Aufruf übergeben.

Ergänzend sei auf `call_user_method_array` hingewiesen, wo Parameter als Array übergeben werden. Das ist interessant, wenn sehr viele Parameter übertragen werden.

Eigenschaften von Klassen ermitteln

Wenn Sie große Projekte in Planung haben, helfen Ihnen einige Funktionen, flexibel auf Klassen zu reagieren. So ermitteln die folgenden Funktionen Eigenschaften von Klassen durch Zugriff auf bereits abgeleitete Objekte:

- `method_exists($object, "method_name")`

Gibt `TRUE` zurück, wenn die Methode in dem angesprochenen Objekt existiert.

- `is_subclass_of($object, "superclass")`

Gibt `TRUE` zurück, wenn das Objekt *\$object* von der Klasse *superclass* abgeleitet wurde, beispielsweise durch `extends`.

- `get_class_methods("classname")`

Diese Funktion gibt ein Array mit allen Methoden der Klasse zurück. Es spielt keine Rolle, ob ein Objekt dieser Klasse erzeugt wurde.

Weitere Funktionen finden Sie in der Referenz. Ein Anwendungsbeispiel zeigt das folgende Skript.

```
<?php
class calc
{
    # Klassendefinition aus Listing 3.87
}
$arr = get_class_methods('calc');
foreach($arr as $method)
    echo "$method<br>";
?>
```

Listing 3.89:
classdefintions:
Ermitteln von
definierten Methoden

Die folgende Abbildung zeigt, welche Methoden definiert wurden.



```
cm2 in
in2 cm
```

Abbildung 3.18:
Ausgabe des Skripts
aus Listing 3.89

Erweiterte OOP-Syntax

PHP 4 hat eine erweiterte und konsequentere Umsetzung der objektorientierten Programmierung. Dies betrifft zum einen Hilfsfunktionen zum Umgang mit Klassen, zum anderen auch die Syntax der Objekte selbst, die nun stärker an C++ angelehnt ist.

```
<?php
class MutterKlasse {

    var $class_name ="Mutterklasse";

    function MutterMethode($value) {
        echo "Muttermethode gibt <b>$value</b> aus.<br>";
    }

    function NamederMethode($string = '') {
        return $string . $this->class_name;
    }
}

class TochterKlasse extends MutterKlasse {

    var $class_name = "Tochterklasse";

    function TochterKlasse($value, $new) {
        MutterKlasse::MutterKlasse($value);
        echo "Tochtermethode gibt <b>$new</b> aus.<br>";
    }

    function TochterMethode($string) {
        return $string . MutterKlasse::NamederMethode();
    }
}
ditrect
```

Listing 3.90:
classdirect: Direkter
Zugriff auf Mutter-
klassen

```
$object = new TochterKlasse("TestMutter", "TestTochter");
echo $object->NamerMethode("Klasse: ");
echo "<br>";
echo $object->TochterMethode("Klasse: ");
?>
```

Wie es funktioniert

Der Zugriff auf diese Klassen erfolgt nun wieder über die schon bekannte Syntax. Zuerst wird ein Objekt der Klasse *Tochterklasse* erzeugt. Sie erbt alle Methoden und Eigenschaften der Klasse *Mutterklasse*. Wenn allerdings der Konstruktor in der Klasse überschrieben wird, geht der Aufruf der übergeordneten Klasse verloren. Der Konstruktor der Mutterklasse wurde hier erweitert (um einen weiteren Parameter), der ursprüngliche Aufruf sollte jedoch erhalten bleiben. Deshalb wird der Konstruktor der *Mutterklasse* direkt aufgerufen – ohne ein Objekt zu instanziiieren:

```
Mutterklasse::MutterKlasse($value);
```

Wird dies nicht gemacht, bleibt die Ausführung allein der entsprechenden Methode der *Tochterklasse* vorbehalten. Bei der Eigenschaft `$class_name` ist dies zu beobachten. Der Wert, der in der *Mutterklasse* zugewiesen wurde, ging mit der Vererbung an die *Tochterklasse* verloren. Löschen Sie die folgende Zuweisung in der *Tochterklasse*, bleibt der alte Wert erhalten:

```
var $class_name = "Tochterklasse";
```

Nun ist diese Darstellung extrem abstrakt, denn die Klasse erfüllt keinen praktischen Zweck. Trotzdem sollten Sie sich die Zeit nehmen, mit den Methoden und verschiedenen Testwerten ein wenig zu spielen, um die Reaktion der Werte zu erkennen.

Abbildung 3.19:
Ausgabe des Skripts
aus Listing 3.90

```
Muttermethode gibt TestMutter aus.
Tochtermethode gibt TestTochter aus.
Klasse: TochterKlasse
Klasse: TochterKlasse
```

Hilfs- und Testfunktionen

Zum Umgang mit Objekten gehören nun einige Hilfsfunktionen. Zuerst eine Übersicht:

```
get_class()
get_parent_class()
method_exists()
class_exists()
is_subclass_of()
get_class()
    _methods()
get_declared
    _classes()
get_class_vars()
get_object_vars()
```

- `get_class`
Ergibt den Namen der Klasse des Objekts.
- `get_parent_class`
Ergibt den Namen der übergeordneten Klasse des Objekts.
- `method_exists`
Prüft, ob eine Methode existiert.

- `class_exists`
Ermittelt, ob die Klasse definiert wurde.
- `is_subclass_of`
Prüft, ob ein Objekt zu einer Unterklasse gehört.
- `get_class_methods`
Gibt ein Array mit den Namen der Methoden einer Klasse zurück.
- `get_declared_classes`
Gibt die Namen aller deklarierten Klassen in einem Array zurück. Zusätzlich zu den selbstdefinierten werden drei interne Klassen ausgegeben: `stdClass`, `OverloadedTestClass`, `Directory` (ab PHP 4.0.1 Patchlevel 2).
- `get_class_vars`
Diese Funktion gibt in einem Array die Namen der Eigenschaften der Klasse zurück.
- `get_object_vars`
Mit dieser Funktion ermitteln Sie die Eigenschaften eines Objekts, also die tatsächlich genutzten Variablen der zugrunde liegenden Klasse.

Die Anwendung ist verhältnismäßig einfach, wie das folgende Beispiel zeigt. Die Werte beziehen sich auf das letzte Beispiel mit den Mutter- und Tochterklassen (zugegeben primitiv):

```
<?php
$objM = new MutterKlasse('M');
$objT = new TochterKlasse('T1', 'T2');
echo '<p>Klassen testen:';
echo '<BR>Wo von stammt Tochterklasse ab? ';
echo get_parent_class($objT);
echo '<BR>Wie heißt die Klasse des Objekts $objM? ';
echo get_class($objM);
echo '<BR>Hat $objM eine Mutterklasse? ';
echo get_parent_class($objM) ? 'JA' : 'NEIN';
echo "<BR>";
echo method_exists($objT,"TochterMethode") ?
    "TochterMethode existiert" :
    "TochterMethode existiert nicht";
echo "<BR>";
echo method_exists($objT,"TochterStart") ?
    "TochterStart existiert" :
    "TochterStart existiert nicht";
echo "<BR>Methoden der Tochterklasse: <BR>";
$arr = get_class_methods('Tochterklasse');
```

Listing 3.91:
checkclass: Ermitteln
von Informationen
über Klassendefini-
tionen


```
foreach ($arr as $m) echo "&nbsp;-&nbsp;$m<br>";
?>
```

Beachten Sie, dass in einigen Fällen der Name der Klasse als Zeichenkette, in anderen die Objektvariable selbst als Parameter anzugeben ist. Mehr Informationen dazu finden Sie im Referenzhandbuch »PHP 4. Die Referenz«, das ebenfalls bei Carl Hanser erschienen ist.

Anwendung zur Fehlersuche

In der praktischen Programmierung werden diese Funktionen sicher weniger benötigt, denn welche Methoden existieren, sollte einem als Programmierer schon klar sein – vor allem, wenn man sie selbst geschrieben hat. Interessanter ist die Anwendung bei der Fehlersuche, wenn es darum geht festzustellen, ob Objekte richtig instanziiert wurden, die Vererbung funktioniert hat oder die Namen stimmen.

Abbildung 3.20:
Verschiedene
Informationen über
Klassen

```
Muttermethode gibt M aus.
Muttermethode gibt T1 aus.
Tochtermethode gibt T2 aus.
Klassen testen:
Wo von stammt Tochterklasse ab? mutterklasse
Wie heißt die Klasse des Objekts $objM? mutterklasse
Hat $objM eine Mutterklasse? NEIN
TochterMethode existiert
TochterStart existiert nicht
Methoden der Tochterklasse:
- mutterklasse
- namedermethode
- tochterklasse
- tochtermethode
```

3.4.5 Praktischer Umgang mit Klassen

Ein Anwendungsbeispiel

In der Praxis werden Klassen überwiegend eingesetzt, um in vielen Programmen wiederverwendet werden zu können. Solche Definitionen zeichnen sich deshalb durch eine gewisse Universalität aus. Das folgende Projekt zeigt die Entwicklung einer einfachen Klasse und eine Strategie, wie davon andere abgeleitet werden können.

Anwendungsbeispiel

Häufig benötigt man beim Aufbau von Webseiten kleine Grafiken, die als Abstandhalter, Linien oder Graphen verwendet werden. Dies sind in der Regel kleine GIF-Bildchen, einfarbige, im Idealfall nur 1x1 Pixel große Elemente, die erst im Browser ihre endgültige Größe erreichen. Nun können Sie sich Dutzende Varianten vorher in Adobe Photoshop anfertigen – oder diese von PHP generieren lassen.

Bilder selbstgemacht

Hier geht es nur um die Erzeugung eines kleinen Standard-GIFs. Die innere Struktur eines solchen Bildes ist klar festgelegt. Entsprechend bietet es sich an, dies in der Klasse bei Bedarf zusammenzusetzen. GIF ist – wie jedes andere Bildformat auch – sehr genau definiert. Praktisch setzt sich ein Bild aus Basisinformationen und Bilddaten zusammen. Diese Informationen werden hier verwendet, um ein GIF-Bild dynamisch zu erzeugen.

Die erste Klasse

Die erste Klasse erzeugt lediglich ein schwarzes Pixel. Sie wird später um eine Farbkomponente erweitert.

```
<?php
class giffy {
    var $ini = '47494638396101000100f70000';
    var $end;
    var $nor = '2c000000000100010000080400010404003b';
    var $inc = '21f90401000000002c000000000
                100010000080400010404003b';
    var $color = '000000';
    function giffy () {
        $this->end = str_repeat('f', 1530);
    }
    function hex2bin($s)
    {
        $n = strlen($s);
        if ($n % 2 != 0) { return FALSE; }

        $t = "";
        for ($x = 0; $x < $n; $x+=2) {
            $t .= chr(hexdec($s[$x].$s[$x+1]));
        }
        return $t;
    }
    function create ($incolor = 0) {
        $pix = $this->ini . $this->color . $this->end;
        if ($incolor == 1)
        {
            $pix .= $this->inc;
        } else {
            $pix .= $this->nor;
        }
        return $this->hex2bin($pix);
    }
}
?>
```

Listing 3.92:
Klasse zum Erzeugen
eines 1x1-Pixel-GIF

Die Klasse definiert folgende intern verwendete Eigenschaften:

- *\$ini*. Startsequenz der GIF-Datei
- *\$end*. Endsequenz der Farbtabelle der GIF-Datei
- *\$nor*. Endsequenz für transparente Bilder
- *\$inc*. Endsequenz für nichttransparente Bilder

Die Variable *\$end* besteht nur aus FF-Bytes und wird mit dem Konstruktor *giffy()* auf 1530 Bytes aufgefüllt. Konstruktoren werden aufgerufen, wenn ein Objekt der Klasse instanziiert wird. Das Bild

Wie es funktioniert

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

selbst wird in der Methode *create* zusammengesetzt. Da mit Hexadezimalzahlen gearbeitet wird, die in Form von Zeichenketten existieren, müssen diese am Ende noch in eine binäre Form gebracht werden. Leider verfügt PHP nicht über eine fertige Funktion dazu. Deshalb wurde *hex2bin* entworfen. Die Funktion nimmt jeweils zwei Zeichen und rechnet sie in einen Dezimalwert um, der wiederum mit *chr* in ein Zeichen umgesetzt wird.

Verwendung der Klasse

So wird es angewendet

Die reihenweise Erzeugung von Bildern ist sicher nicht besonders intelligent. Einfacher ist es, die generierten Bilder direkt zum Browser zu senden. Die Klasse wird deshalb in eine Datei eingebaut, die ein Bild erzeugt und direkt an den Browser sendet.

Listing 3.93:
sendgif: Senden eines Bildes zum Browser

```
<?php
class giffy
{
    # Klassendefinition aus Listing 3.92
}
$gifpix = new giffy();
header("Content-type: image/gif");
echo $gifpix->create();
?>
```

Wie es funktioniert

Das Objekt wird erzeugt und mit *echo* an den Browser gesendet. Damit dieser die Angaben bearbeiten kann, werden entsprechende Kopfdaten mit *header* erzeugt – hier der MIME-Typ *IMAGE/GIF*. Wichtig ist, dass vor der Ausgabe von Headern keine weiteren Daten gesendet werden – auch keine Leerzeichen.

Natürlich muss der Aufruf des Skripts aus Listing 3.93 irgendwie initiiert werden. Dazu dient eine kleine HTML-Datei als Test:

Listing 3.94:
usegif: Nutzung der dynamischen GIF-Erzeugung

```
<table>
<tr><td rowspan="5">
    
</td></tr>
<tr><td>
    
</td></tr>
<tr><td>
    
</td></tr>
<tr><td>
    
</td></tr>
<tr><td>
    
</td></tr>
</table>
```

Das Skript besteht nur aus reinem HTML. Beachten Sie den Aufbau der ``-Tags. Hier wird als Ziel kein Bild angegeben, sondern das zuvor definierte PHP-Skript.

Wie es funktioniert

Das Ergebnis ist sicher schon recht ansehnlich. Allerdings macht dies noch wenig Sinn, denn ein schwarzes GIF abzulegen wäre einfacher gewesen.

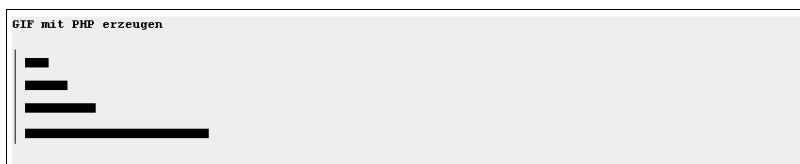


Abbildung 3.21:
Test der Klasse
»giffy«

Eine Erweiterung der Klasse um eine Farbfunktion ist deshalb der nächste Schritt.

Erweiterung einer Klasse

Die Erweiterung erfolgt mit Hilfe der Anweisung `extends`. Dabei kann eine Erweiterung der Klasse nicht nur die Eigenschaften und Methoden der Mutterklasse erben, sondern auch überschreiben. Beides wird in der folgenden Definition ausgenutzt.

```
<?php
class colgif extends giffy
{
    function create($color, $incolor = 0)
    {
        $this->color = $color;
        $gif = giffy::create($incolor);
        return $gif;
    }
}
?>
```

Listing 3.95:
extendgif:
Erweiterung der
Mutterklasse

Nach der Einleitung mit `extends` wird eine neue Methode `create` definiert. Diese überschreibt also die bereits in der Mutterklasse vorhandene Funktion. Neu an der Definition ist ein weiterer Parameter: `$color`. Dieser wird verwendet, um die entsprechende Eigenschaft in der Mutterklasse zu setzen. Das sieht trickreich aus, ist aber ein typischer Vorgang: Alle Eigenschaften der Mutterklasse stehen auch in der abgeleiteten Klasse zur Verfügung. Dann wird die alte Funktion `create` der Mutterklasse aufgerufen. Die Klasse wird hier direkt adressiert und mit dem `::`-Operator angesprochen, weil die Verwendung von `$this` zu einem Namenskonflikt führt. Die Parameter entsprechen natürlich der alten Definition.

Wie es funktioniert

Die Verwendung unterscheidet sich nur durch den zusätzlichen Parameter `$color` und natürlich durch Aufruf der neuen Klasse:

Die Klasse verwenden

```
<?php
$gifpix = new colgif();
header("Content-type: image/gif");
echo $gifpix->create($color);
?>
```

Die Testdatei wird ebenfalls erweitert. Dazu wird der Parameter *color* an den Aufruf des Skripts angehängt:

```

```

*Listing 3.96:
usecgif:
Aufruf der farbigen
GIFs (Ausschnitt)*

Auch wenn es der Druck nicht erlaubt: Abbildung 3.22 zeigt farbige Bilder. Von Vorteil ist nun, dass Sie die Farben dynamisch erzeugen können, beispielsweise für eine personalisierte Seite, bei der sich die Benutzer aus einer Vielzahl von Farben eigene aussuchen können. Sie sparen sich die prophylaktische Anfertigung hunderter Minibildchen.

*Abbildung 3.22:
Tatsächlich farbig:
Bunte GIFs
dynamisch erzeugt
(hier als Graustufen)*



Fehlersuche

Wenn es nicht funktioniert

Nachteilig bei diesem Verfahren sind die mangelnden Testmöglichkeiten. Fehler im Skript werden nicht angezeigt, weil der Browser fehlerhafte Bilddaten – die eine Fehlerausgabe für ihn ist – einfach ignoriert oder ein Ersatzsymbol darstellt. Rufen Sie die Klassendefinition deshalb zuerst direkt auf. Dann sehen Sie zwar kein Bild, dafür aber eventuell auftretende Warnungen oder Hinweise. Wenn nichts mehr angezeigt wird, handelt es sich um Bilddaten – dann steht dem regulären Aufruf nichts mehr im Wege.

3.4.6 COM-Objekte



Neu in PHP 4 sind Funktionen zur Programmierung der COM- oder DCOM-Schnittstelle. Um es vorwegzunehmen: COM (Component Object Model) und DCOM (Distributed COM) sind auf die Microsoft-Welt beschränkt und stellen das Gegenstück zu CORBA dar. Eine CORBA-Implementierung war zum Zeitpunkt der Drucklegung in PHP 4 nicht implementiert. In Zukunft ist eine Unterstützung geplant.



Wenn Sie nur mit Linux/Unix arbeiten, können Sie diesen Abschnitt überspringen, sämtliche Anwendungen und Beispiele beziehen sich auf eine Windows-Umgebung. Der Umstand, dass zuerst COM und nicht CORBA implementiert wurde, ist vor allem in der starken Konkurrenz zu ASP zu suchen, wo die COM-Unterstützung praktisch

»von Hause aus« enthalten ist. Vor allem für die Portierung von ASP nach PHP ist COM wichtiger und sinnvoller als CORBA.

Portierungen der COM-Welt nach Unix sind von vielen Stellen geplant und angekündigt. Informieren Sie sich auf den Linux-Seiten über die entsprechenden Fortschritte, beispielsweise unter der Adresse:

<http://www.slashdot.org>

Was ist COM und DCOM?

Um COM zu verstehen, muss man sich ein wenig mit der Windows-Umgebung auseinandersetzen. Die Überlegung, die hinter COM steckt, ist die gleiche, die auch zur Entwicklung der objektorientierten Programmierung führte. Dort wurde angestrebt, leicht wiederverwendbaren Code zu schreiben, um größere Softwareprojekte überhaupt in einer akzeptablen Zeit erstellen zu können. C++ als die Standardprogrammiersprache für Software verfügt über einen ausgeklügelten Mechanismus zur Programmierung von Objekten. Nachteilig ist, dass diese Art der Programmierung nur auf Quelltextebene zur Verfügung steht. Um ein C-Objekt in einer anderen Applikation verwenden zu können, muss es als Quellcode eingebunden und neu übersetzt werden. Das ist zwar verhältnismäßig praktikabel, legt aber jede neue Applikation auf eine bestimmte Programmierungsumgebung fest.

Hinter COM steckt nun die Überlegung, eine solche Distribution von Objekten auf binärer Ebene durchzuführen. Nachteilig ist dabei die Festlegung auf die Plattform, Binärdateien für Intel/Windows lassen sich kaum auf Sparc/Solaris zum Laufen bringen. COM verfügt allerdings über Schnittstellen, die eine Portierung prinzipiell erleichtern. Eine inzwischen fast klassische Anwendung sind die ActiveX-Steuerelemente, die im Browser ebenso laufen wie unter Visual C++, Visual Basic und eben auch unter Delphi und vielen anderen Entwicklungsumgebungen. Eine erneute Übersetzung ist nicht notwendig und Programmierern mit vergleichsweise primitiver Ausstattung – sprich Visual Basic – steht die Funktionalität von COM-Komponenten zur Verfügung, die in C++ geschrieben wurden. Dies alles wohlgedacht ohne die Notwendigkeit, einen C-Compiler bedienen zu müssen. Dies mag Unix-Programmierern, die C praktisch mit der Muttermilch aufgesogen haben, zwar nicht als nennenswerter Vorteil erscheinen, hat sich in der Praxis aber rasend schnell verbreitet.

Eine der Grundideen verteilter Objekte ist auch die Verfügbarmachung innerhalb eines Netzwerks. In COM ist dies einfach durch die Erweiterung DCOM realisiert worden. DCOM erlaubt es, Objekte von entfernten Servern aus aufzurufen und zu starten.



An dieser Stelle wird zumindest klar, wozu COM überhaupt dient. »Das kann aber auch Java«, werden einige jetzt denken. Richtig, Java erstellt Objekte, die sich als Java-Code ohne Betrachtung des Quellcodes verwenden lassen. Aber es hat einen entscheidenden Nachteil: Mit dem Gewinn der Plattformunabhängigkeit wurde wieder die Bindung an die Entwicklungsumgebung eingekauft – Java oder JavaBeans laufen nur unter Java. Zwar gibt es Schnittstellen und Brücken, aber die machen die so erstellte Software nicht unbedingt schnell.

Dagegen scheinen sich Portierungen von COM auf Unix schneller zu verbreiten. Die neuesten Informationen dazu finden Sie unter der Adresse <http://www.microsoft.com/com/>. Einige Softwarehersteller haben eigene Schnittstellen für verschiedene Betriebssysteme geschrieben, beispielsweise ist DCOM von der Software AG für HP-UX, Solaris, Digital UNIX, OS/390 Unix und AIX zu bekommen.

Beispiel

Um Verbindungen zu Programmen per COM herzustellen, benötigt man natürlich eine entsprechende Applikation. Ein gutes Beispiel ist der Zugriff auf ADO, die Datenbankabstraktionsebene unter Windows. VBScript-Programmierer werden den folgenden Code sicher leicht lesen können.

*Listing 3.97:
com_withado:
ADO über COM-
Funktionen von PHP
verwenden*

```
<?php
$objConn = new COM("ADODB.Connection") or
    die("ADO nicht verfügbar");
$objConn->Open("DSN=InetLog");
$objRS = $objConn->Execute("SELECT * FROM InetLog");
$intFields = $objRS->Fields->Count();
echo '<table border="1"><tr bgcolor="#eeeeee">';
for ($i=0; $i < $intFields; $i++)
{
    $fld[$i] = $objRS->Fields($i);
    echo '<th>' . $fld[$i]->name . '</th>';
}
while (!$objRS->EOF)
{
    echo '<tr>';
    for ($i=0; $i < $intFields; $i++)
    {
        echo '<td>' . $fld[$i]->value . '</td>';
    }
    echo '</tr>';
    $objRS->MoveNext();
}
echo '</tr></table>';
$objRS->Close();
$objConn->Close();
?>
```

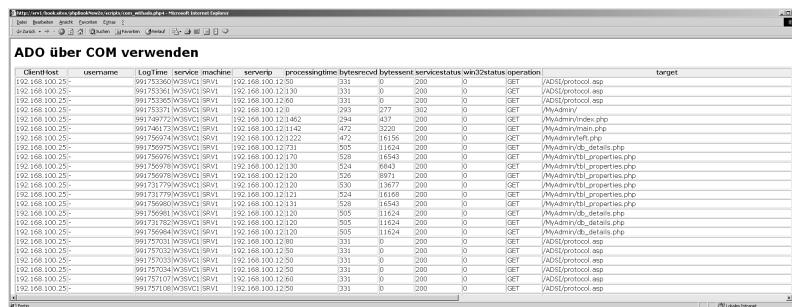
Der eigentliche Start einer COM-Instanz erfolgt in der ersten Zeile – hier durch Aufruf der ADO-Klasse Connection. Dann wird eine ODBC-Verbindung aufgebaut, wobei hinter Open alles erlaubt ist, was ADO auch sonst akzeptiert. Beachten Sie, dass der Zugriff auf die Varianten, wie sie VBScript erlaubt, nicht direkt funktioniert. Deshalb werden die Objekte zuerst einer Variablen zugewiesen:

```
$fld[$i] = $objRS->Fields($i);
```

Dann erst, im zweiten Schritt, kann auf die Eigenschaften zugegriffen werden:

```
echo '<th>' . $fld[$i]->name . '</th>';
```

Das Skript setzt eine Tabelle voraus, in der folgenden Abbildung wird eine Protokolldatei eines Webservers angezeigt.



ClientHost	username	LogTime	service	machine	serverip	processingtime	bytesreceived	bytesresent	servicestatus	win32status	operation	target
192.168.100.25		991751360	W3SVC1	SRV1	192.168.100.12.50	331	0	200	0	0	GET	ADSLprotocol.asp
192.168.100.25		991751361	W3SVC1	SRV1	192.168.100.12.130	331	0	200	0	0	GET	ADSLprotocol.asp
192.168.100.25		991751365	W3SVC1	SRV1	192.168.100.12.80	331	0	200	0	0	GET	ADSLprotocol.asp
192.168.100.25		991751371	W3SVC1	SRV1	192.168.100.12.0	243	277	302	0	0	GET	AdminV
192.168.100.25		991749772	W3SVC1	SRV1	192.168.100.12.1403	294	437	200	0	0	GET	AdminIndex.php
192.168.100.25		991748175	W3SVC1	SRV1	192.168.100.12.1142	472	3225	200	0	0	GET	AdminIndex.php
192.168.100.25		991756974	W3SVC1	SRV1	192.168.100.12.1222	402	16156	200	0	0	GET	AdminIndex.php
192.168.100.25		991756975	W3SVC1	SRV1	192.168.100.12.731	505	11624	200	0	0	GET	AdminIndex_details.php
192.168.100.25		991756976	W3SVC1	SRV1	192.168.100.12.170	528	16543	200	0	0	GET	AdminIndex_properties.php
192.168.100.25		991756978	W3SVC1	SRV1	192.168.100.12.130	524	6843	200	0	0	GET	AdminIndex_properties.php
192.168.100.25		991756978	W3SVC1	SRV1	192.168.100.12.120	528	8971	200	0	0	GET	AdminIndex_properties.php
192.168.100.25		991751739	W3SVC1	SRV1	192.168.100.12.120	530	13677	200	0	0	GET	AdminIndex_properties.php
192.168.100.25		991751739	W3SVC1	SRV1	192.168.100.12.121	524	11668	200	0	0	GET	AdminIndex_properties.php
192.168.100.25		991756980	W3SVC1	SRV1	192.168.100.12.131	528	16543	200	0	0	GET	AdminIndex_properties.php
192.168.100.25		991756981	W3SVC1	SRV1	192.168.100.12.120	505	11624	200	0	0	GET	AdminIndex_details.php
192.168.100.25		991751782	W3SVC1	SRV1	192.168.100.12.120	505	11624	200	0	0	GET	AdminIndex_details.php
192.168.100.25		991756984	W3SVC1	SRV1	192.168.100.12.120	505	11624	200	0	0	GET	AdminIndex_details.php
192.168.100.25		991757031	W3SVC1	SRV1	192.168.100.12.80	331	0	200	0	0	GET	ADSLprotocol.asp
192.168.100.25		991757032	W3SVC1	SRV1	192.168.100.12.50	331	0	200	0	0	GET	ADSLprotocol.asp
192.168.100.25		991757033	W3SVC1	SRV1	192.168.100.12.50	331	0	200	0	0	GET	ADSLprotocol.asp
192.168.100.25		991757034	W3SVC1	SRV1	192.168.100.12.50	331	0	200	0	0	GET	ADSLprotocol.asp
192.168.100.25		991757107	W3SVC1	SRV1	192.168.100.12.60	331	0	200	0	0	GET	ADSLprotocol.asp
192.168.100.25		991757108	W3SVC1	SRV1	192.168.100.12.50	331	0	200	0	0	GET	ADSLprotocol.asp

Abbildung 3.23:
Zugriff auf Access
per ADO mit PHP

Wenn Sie an dieser Stelle den Einsatz dieser Funktion für den Zugriff auf Excel-Tabellen erwägen, ist ein Hinweis auf einen anderen Lösungsweg angebracht: Excel verfügt über einen ODBC¹¹-Treiber. Sie können einfacher und bequemer mit den Datenbankfunktionen auf solche strukturierten Datenquellen zugreifen.



3.5 Fehlerbehandlung

Kein Skript ohne Fehler – diese Erfahrung werden Sie schnell machen. Sie erfahren hier, wie Sie mit Fehlern umgehen und mit welchen Methoden Sie Ihre Skripte von Fehlern befreien.

3.5.1 Abbruchsteuerung

Im Zusammenhang mit der allgemeinen Fehlerbehandlung sind einige Sprachkonstrukte einsetzbar. Die einfachste Möglichkeit, auf Fehler zu reagieren, besteht in der Beendigung der Skriptausführung. Zwei Funktionen können Sie einsetzen:

¹¹ Informationen über ODBC finden Sie in Kapitel 6.

- die
- exit

die

Der Befehl `die` beendet das laufende Skript, gibt aber zuvor noch eine Meldung an den Browser aus:

```
if ($status="error") {
    die("Ein Fehler ist aufgetreten");
}
```



Eine interessante Anwendung für `die` ist die syntaktische Kopplung an andere Funktionen, beispielsweise mit dem Operator `or`:

```
<?php
$filename = '/path/to/data-file';
$file = fopen($filename, 'r') or die("Fehler bei: ".$filename);
?>
```

Wenn die Funktion `fopen` nicht korrekt abgearbeitet wird, bricht das Skript hier mit der Meldung ab. Die Funktionsweise hat übrigens nichts mit `die` zu tun, sondern reflektiert nur die Arbeitsweise des Ausdrucksparsers. Ausdrücke werden von links nach rechts abgearbeitet. Wenn der Ausdruck ein eindeutiges Ergebnis hat, bricht der Parser ab und ignoriert den Rest der Zeile. Bei Oder (`or`) bedeutet dies, dass der Ausdruck in jedem Fall `TRUE` wird, wenn der links stehende Teil `TRUE` ist. Der rechte Teil (die "Message") wird also überhaupt nicht angefasst. Gibt dagegen die links stehende Funktion `FALSE` zurück, wird der Parser gezwungen auch den rechten Teil abzuarbeiten; `die` wird ausgeführt. Die Funktion `die` gibt selbst nichts zurück.

exit

Ähnlich funktioniert auch die Funktion `exit`. Allerdings wird keine Meldung ausgegeben, der Abbruch erfolgt sofort.

```
if ($checkit) { exit; }
```

3.5.2 Das Fehlerkonzept in PHP

PHP unterscheidet vier praktisch nutzbare Klassen von Fehlern unterschiedlicher Schwere:

- (1) Normale Fehler in Funktionen
- (2) Normale Warnungen
- (4) Parserfehler
- (8) Nachrichten

Daneben gibt es weitere Klassen, die auf nutzerdefinierte Ereignisse reagieren.

php.ini

Die Fehler werden als Bitfeld zusammengefasst und ergeben einen Fehlerstatus. Ausgaben erfolgen nur dann an den Browser, wenn das

entsprechende Fehlerniveau erreicht wird. Der Standardwert ist 7, also $1 + 2 + 4$. Damit werden alle Fehler ausgegeben, außer Nachrichten. Sie können die Einstellung in der Datei `PHP.INI` mit dem folgenden Eintrag ändern:

```
php_error_reporting = 7
```

Darüber hinaus werden interne Fehler (so genannte Kernel¹²-Fehler) erkannt und mit der Funktion `error_reporting` ausgegeben. Die folgende Tabelle zeigt alle Bitwerte und die Namen der Fehlerklassen:

error_reporting()

Bitwert	Fehlername	Beschreibung
1	E_ERROR	Fehler im Programmablauf
2	E_WARNING	Warnungen
4	E_PARSE	Fehler in der Syntax
8	E_NOTICE	Nachrichten
16	E_CORE_ERROR	Fehler des PHP-Kernels
32	E_CORE_WARNING	Warnung des PHP-Kernels
64	E_COMPILE_ERROR	Zend-Compiler-Fehler
128	E_COMPILE_WARNING	Zend-Compiler-Warnung
256	E_USER_ERROR	Diese Konstanten steuern das Verhalten der nutzerdefinierten Fehlerbehandlung mit der Funktion <code>user_error</code>
512	E_USER_WARNING	
1024	E_USER_NOTICE	
	E_ALL	Alle Fehler

Tabelle 3.16:
Fehlerklassen der
Funktion
`error_reporting`

Die Fehlernamen sind in PHP als Konstanten vordefiniert.

3.5.3 Umgang mit Laufzeitfehlern

Es ist sinnvoll, auf Fehler, die der Interpreter erzeugt, während der Laufzeit gezielt zu reagieren. Dies erspart oft viele zusätzliche Prüfungen und Abfragen, die ihrerseits wieder Fehlerquellen darstellen.

Fehler abfangen

Oft werden minder schwere Fehler ausgegeben, die auf unsichere Funktionsauswertungen oder fehlende Parameter zurückzuführen sind. Grundsätzlich sollten Ihre Skripte natürlich fehlerfrei arbeiten, manchmal treten Fehler aber nur selten auf oder werden bewusst in

¹² Als Kernel (engl. Core) wird jener Teil des PHP-Interpreters bezeichnet, der die Kernfunktionen und Sprachkonstrukte realisiert.

Listing 3.98:
*noerror: Fehler
unterdrücken*

Kauf genommen. Sie können solche Fehler unterdrücken, indem vor den Aufruf einer beliebigen Funktion das Zeichen @ gestellt wird:

```
if (!@fopen('xxx', 'r')) {
    echo "Datei 'xxx' nicht gefunden.";
} else {
    echo "Ausführung ok.";
}
```

In der Regel gibt die Funktion bei fehlerhaftem Aufruf FALSE zurück. Die Auswertung kann deshalb gut mit if-Anweisungen erfolgen.



Hinweis

Bei der Unterdrückung von Funktionsaufrufen nicht vorhandener Funktionen wird der gesamte if-Block nicht ausgeführt. Dieses Verhalten mag je nach Version unterschiedlich ausfallen. Zumindest ist es sinnvoll, Skripte erst auszutesten und dann an kritischen Stellen Systemfehler zu unterdrücken. Es ist im Sinne einer professionellen Programmierung generell nicht empfehlenswert, dem System die Fehlerbehandlung zu überlassen. Dies sollten Sie selbst erledigen: Im letzten Beispiel wäre der Dateitest mit `file_exists` sicher besser.

Fehler auswerten

\$PHP_ERRORMSG Die letzte Fehlermeldung wird in der globalen Variablen `$PHP_ERRORMSG` gespeichert. Werden Fehler unterdrückt, können Sie hier trotzdem die Meldungen auslesen.

php.ini Voraussetzung ist, dass in der Datei `PHP.INI` der folgende Eintrag steht:

```
track_error = True
```

Die Fehleranzeige modifizieren

php.ini In der Konfigurationsdatei können Sie das Format der Fehlerausgabe einstellen. Erlaubt ist jede Art von HTML-Tag. Die Ausgabe erfolgt ohne weitere Bearbeitung. Die folgende Form schreibt alle Fehlermeldungen in Rot:

```
error_prepend_string = "<font color=ff0000>"
error_append_string = "</font>"
```

Vergessen Sie nicht, die Wirkung des Befehls wieder aufzuheben, sonst werden nachfolgende Texte ebenfalls formatiert.

3.5.4 Benutzerdefiniertes Fehlermanagement

PHP 4 verfügt über einige Funktionen, mit denen sich ein eigenes Fehlermanagement aufbauen lässt. PHP verfügt über einen sogenannten »Errorhandler«. Der lässt mit der Funktion `set_error_handler` austauschen. Als Argument ist der Name einer eigenen Funktion anzugeben:

```
set_error_handler('ls_errorhandler');
```

```
set_error_handler()
```

Ab diesem Punkt sind Sie selbst für die Ausgabe aller Fehlermeldungen zuständig. Im Teil IV des Buches wird das umfangreiche Projekt *phpTemple* vorgestellt, das eine eigene Fehlerverwaltung enthält. Daraus stammt folgende benutzerdefinierte Fehlerverwaltung:

```
<?php
function ls_errorhandler($errno, $errstr, $errfile, $errline)
{
    switch ($errno)
    {
        case E_USER_ERROR:
            echo "<span style='color:red'><b>FEHLER [$errno]</b><br></span><br>\n";
            echo "Ausgelöst auf Zeile <b>$errline</b> in Skript <b>$errfile</b><br>\n";
            echo "PHP ".PHP_VERSION." (".PHP_OS.")<br>\n";
            echo "Programmabbruch...<br>\n";
            exit -1;
            break;
        case E_USER_WARNING:
            echo "<span style='color:red'><b>WARNUNG [$errno]</b><br></span><br>\n";
            echo "Skript <b>$errfile</b>, Zeile <b>$errline</b><p></span>\n";
            break;
        case E_USER_NOTICE:
            echo "<span style='color:red'><b>HINWEIS [$errno]</b><br></span><br>\n";
            echo "Skript <b>$errfile</b>, Zeile <b>$errline</b><p></span>\n";
            break;
        case E_NOTICE:
            echo "<span style='color:red'><b>PHP-Nachricht [$errno]</b> $errstr<br>\n";
            echo "Skript <b>$errfile</b>, Zeile <b>$errline</b><p></span>\n";
            break;
        case E_PARSE:
            echo "<span style='color:red'><b>PHP-Parserfehler [$errno]</b> $errstr<br>\n";
            echo "Skript <b>$errfile</b>, Zeile <b>$errline</b><p></span>\n";
            break;
        case E_WARNING:
            echo "<span style='color:red'><b>PHP-Warnung [$errno]</b> $errstr<br>\n";
            echo "Skript <b>$errfile</b>, Zeile <b>$errline</b><p></span>\n";
            break;
        case E_ERROR:
```

Listing 3.99:
usererror:
Beispiel für eine
eigene Fehler-
behandlung

```

        echo "<span style=\"color:red\"><b>PHP-FEHLER [\$errno]</b>
            \$errstr<br>\n";
        echo "Skript <b>\$errfile</b>, Zeile
            <b>\$errline</b><p></span>\n";
        break;
    }
} /* end function */
error_reporting(E_ALL);
set_error_handler('ls_errorhandler');
?>

```

Treten nun innerhalb dieses Skripts Fehler auf, reagiert die Funktion *ls_errorhandler* entsprechend. Der folgende Abschnitt provoziert solche Fehler (Fortsetzung des Skripts aus Listing 3.99):

```

<?php
echo "Start der eigenen Fehlerfunktion:<br>";
echo $cc;
fopen('llk', 'r');
$cc = 0;
if ($cc == 0) {
    trigger_error('0 ist für $cc nicht erlaubt', E_USER_NOTICE);
}
?>

```

Abbildung 3.24:
Selbstdefinierte
Fehlermeldungen

```

Start der eigenen Fehlerfunktion:
PHP-Nachricht [8] Undefined variable: cc

Skript c:\inetpub\wwwroot\book.sites\phpbooknew2e\process_scripts.php4(205) : eval()'d code, Zeile 43

PHP-Warnung [2] fopen("llk","r") - No such file or directory

Skript c:\inetpub\wwwroot\book.sites\phpbooknew2e\process_scripts.php4(205) : eval()'d code, Zeile 44

HINWEIS [1024] 0 ist für $cc nicht erlaubt

Skript c:\inetpub\wwwroot\book.sites\phpbooknew2e\process_scripts.php4(205) : eval()'d code, Zeile 47

```

Obwohl im Handbuch nichts gegenteiliges dazu ausgeführt ist, scheint es auch mit der eigenen Fehlerbehandlung unmöglich zu sein, Parserfehler zu unterdrücken. Dieses Verhalten ist allerdings nachvollziehbar, weil Parserfehler die Funktionsfähigkeit des Parsers insgesamt in Frage stellen und damit auch die eigenen Fehlerbehandlungsfunktionen.

Fehlerbehandlung mit Testfunktionen

assert()
assert_options()

Fehler treten oft erst unter bestimmten Umgebungsbedingungen auf. PHP 4 kennt Funktionen, mit denen ein Test ausgeführt werden kann, ohne dass ein Laufzeitfehler auftritt. Um robuste Skripte zu schreiben, sollten Sie derartige Bedingungen schon bei der Entwicklung simulieren. Die Schwierigkeit besteht meist darin, die Fehler koordiniert abzufangen und die Testbedingungen ein- und auszuschalten. Elegant können Sie dieses Problem mit der Funktion *assert* lösen. Diese Funktion testet eine Bedingung und gibt FALSE zurück, wenn ein Laufzeitfehler aufgetreten wäre.

Das folgende Beispiel ist eine Ergänzung der bereits in Listing 3.99 gezeigten Funktion.

```
<?php
# Einbindung der Funktion ls_errorhandler() aus Listing 3.99
error_reporting(255);
set_error_handler('ls_errorhandler');
echo "Start der eigenen Fehlerfunktion:<br>";
$cc = 100;
$dd = 0;
assert_options(ASSERT_ACTIVE, 1);
assert_options(ASSERT_QUIET_EVAL, 1);
if (!assert($cc / $dd)) {
    trigger_error('Division durch 0', E_USER_WARNING);
}
?>
```

Listing 3.100:
*assert: Anwendung
der Fehlerkontrolle*

Mit der Option `ASSERT_ACTIVE` wird die Funktion `assert` aktiviert oder deaktiviert. Das können Sie also gut zentral steuern. Es ist dann sinnvoll, die Warnungen zu unterdrücken. Im Beispiel wird eine Division durch 0 provoziert. Das löst – egal ob mit dem eigenen oder dem eingebauten Errorhandler – drei Fehler aus:

1. Eine Warnung, dass die Division misslang.
2. Eine weitere Warnung, dass die `assert`-Funktion fehlschlug.
3. Die eigene, durch `assert` provozierte Fehlerbehandlung mit `trigger_error`.

Jetzt lässt sich gut zentral steuern, ob die Warnungen unterdrückt werden und kritische Punkte über `assert` eigene Fehler auslösen.

```
Start der eigenen Fehlerfunktion:
PHP-Warnung [2] Division by zero

Skript c:\inetpub\wwwroot\book.sites\phpbooknew2e\process_scripts.php4(205) : eval()'d code, Zeile 50
PHP-Warnung [2] Assertion failed

Skript c:\inetpub\wwwroot\book.sites\phpbooknew2e\process_scripts.php4(205) : eval()'d code, Zeile 50
WARNUNG [512] Division durch 0

Skript c:\inetpub\wwwroot\book.sites\phpbooknew2e\process_scripts.php4(205) : eval()'d code, Zeile 50
```

Abbildung 3.25:
*Fehlermeldungen der
Testfunktionen*

Wenn Sie generell die Warnungen unterdrücken und nicht mit `assert` arbeiten, können Sie kritische Fehlerstellen nicht mehr explizit erkennen. Wenn Sie jeden theoretisch möglichen Zustand im Skript verarbeiten (was nicht immer wirklich möglich ist), ist `assert` natürlich obsolet, denn ein darüber hinaus gehender »Ausnahmestandard« existiert nicht mehr.

3.5.5 Fehler vermeiden

Fehler in Skripten sind nie ganz zu vermeiden. Nur sehr einfacher Code kann perfekt sein. Trotzdem können Sie mit einer Reihe von Maßnahmen die Qualität der Skripte erhöhen. Diese freiwillige Leistung des Programmierers erhöht die Qualität und trägt zu einer schnelleren Umsetzung des Projekts bei, auch wenn es an einigen Stellen einen höheren Schreibaufwand erfordert. Zu den wichtigsten Maßnahmen gehören:

- Einhaltung von Code-Konventionen
- Formatierung und Strukturierung
- Abfrage von Zuständen

Einhaltung von Code-Konventionen

Code-Konventionen sind (aus Sicht des PHP-Parsers) unverbindliche Vorschriften oder Empfehlungen zur Schreibweise von Variablen und Funktionen. Grundsätzlich müssen die Namen nur den Anforderungen der Semantik genügen. Für die Lesbarkeit und Fehlersuche sind aber erweiterte Benennungsregeln empfehlenswert.

Neben den vielen Vorschriften, die eine Programmier- oder Skriptsprache ausmachen, gibt es auch eine Reihe von Empfehlungen, die Programmierern helfen, gut lesbare und saubere Programme zu schreiben. Die Code-Konventionen sind solche Empfehlungen. Dabei geht es auch um die Pflege der Programme durch andere Programmierer oder später durch Sie selbst. Oft entstehen Programme unter großem Zeitdruck, die nötige Kommentierung und Dokumentation wird nur mangelhaft möglich sein. Die für jede Sprache geltenden Konventionen erleichtern dann das Einarbeiten in fremdem oder eigenen Code.

Trotzdem gibt es keine Prüfung oder keinen Zwang wie bei der Sprachsyntax, solche Empfehlungen einzuhalten. Es ist eine freiwillige Leistung der Programmierer, guten Code zu schreiben. Die Konventionen umfassen dabei die folgenden Bereiche:

- Namenskonventionen für Variable, Objekte und Prozeduren,
- Kommentarkonventionen sowie
- Textformatierung und Einrückungen.

Die hier wiedergegebenen Empfehlungen entsprechen weitestgehend den Vorgaben der Microsoft-Guidelines zu VBScript, die in dieser Form für PHP leider noch nicht existieren.

Konstanten kennzeichnen

Konstanten können einfach durch Großbuchstaben kenntlich gemacht werden:

```
MEINE_KONSTANTE
```

```
FARBE_FRAME_ROT
```

Bestehen die Namen aus mehreren Wörtern, wurden die Wörter mit dem Unterstrich »_« getrennt und alle Buchstaben groß geschrieben. Wem es besser gefällt, der kann auch ein Präfix zur Kennzeichnung davor setzen:

```
conMeineKonstante
```

```
conFarbeFrameRot
```

Bei längeren Bezeichnern werden die Anfangsbuchstaben der Wörter groß geschrieben. Zu lange Namen sind nicht empfehlenswert, da der Schreibaufwand bei häufiger Verwendung immens steigt und die Lesbarkeit durch lange Codezeilen nicht unbedingt steigt. Da Speicherplatz aber normalerweise keine Rolle spielt, gibt es keinen Grund für kryptische Abkürzungen.

Variablen werden wie die neuen Konventionen für Konstanten mit vorgestellten Typcodes geschrieben. Die folgende Tabelle zeigt die Empfehlung und ein Beispiel.

Variablen und Datentypen

Datentyp	Präfix	Beispiel
Array	arr	\$arrRS, \$arrNames
Boolean	bln	\$blnTest
Byte	byt	\$bytPixelWert, \$bytPixValue
Date (Time)	dt	\$dtStartWert, \$dtStartDate
Double	dbl	\$dblErgebnis, \$dblResult
Error	err	\$errNummer, \$errNumber
Integer	int	\$intMenge, \$intQuantity
Long	lng	\$lngAbstand, \$lngDistance
Object	obj	\$objAktuell, \$objCurrent
Single	sng	\$sngMittelwert, \$sngAverage
String	str	\$strVorName, \$strName

Tabelle 3.17:
Vorschlag für die
Kennzeichnung der
Datentypen bei
Variablen

Ein Wort zu deutschen oder englischen Variablenbezeichnern. Wenn Sie und Ihre Mitarbeiter gut Englisch können, sind Sie mit englischen Bezeichnern besser beraten. Die Worte sind in der Regel kürzer, prägnanter und einfacher. In allen anderen Fällen können Sie auch deutsche Bezeichner verwenden, welche die Lesbarkeit verbessern, denn gute Variablenbezeichner machen einen Quellcode natürlichsprachig. Beachten Sie aber, dass nie Umlaute verwendet werden dürfen und

Deutsch oder Englisch?

dass ungelenke Übersetzungen computertypischer englischer Begriffe oft eher verwirrend wirken.

Geltungsbereich der Variablen

Variablen sollten mit dem geringst möglichen Geltungsbereich definiert werden. In PHP kann man praktisch nur zwischen globalen und lokalen Variablen unterscheiden. Lokale Variablen gelten dabei nur innerhalb der Struktur, in der sie definiert wurden. Normalerweise ist dies eine Prozedur oder Funktion. Wenn Variablen global sind, sollten sie im HEAD-Sektor der HTML-(PHP)-Datei definiert werden. Um globale von lokalen Variablen anhand des Namens unterscheiden zu können, wird der Buchstabe »s« (für Skript) vorangestellt.

Funktionen und Objekte benennen

Wie schon bei den Variablen wird auch für Prozeduren und Funktionen ein möglichst vollständiger, beschreibender Name mit mehreren Wörtern benutzt. Die Anfangsbuchstaben der Wörter werden großgeschrieben. Da Funktionen immer irgendeine Aufgabe ausführen, sollten sie mit einem Verb beginnen:

```
SchliesseDatei  
InitArray
```

Deutsch oder Englisch wurde bereits behandelt. Auch hier ist die Einheitlichkeit oberstes Gebot! Verwenden Sie keine englischen Variablen und keine deutsche Funktionsnamen.

Kommentare richtig nutzen

Mit Kommentaren wird Quelltext lesbarer und für die spätere Weiterbearbeitung wie auch bei der Fehlersuche leichter handhabbar. Kommentare in PHP werden nicht an den Browser weitergeleitet, sodass Sie unbesorgt auch serverseitige Details beschreiben können. Lediglich echte HTML-Kommentare gelangen zum Nutzer. Dort wird dann der Name des Autors oder Betreibers übermittelt.

Jede Prozedur sollte mit einer kurzen Zweckbeschreibung beginnen. Schreiben Sie immer, wozu eine Prozedur implementiert wurde, und nicht, wie sie arbeitet. Nennen Sie weiter alle globalen Variablen, Objekte oder anderen Elemente, die Sie innerhalb der Prozedur nutzen. Nennen Sie auch jede externe Variable oder jedes externe Objekt, das innerhalb der Prozedur geändert wird. Beschreiben Sie Struktur und Wertebereich der Rückgabewerte bei Funktionen.

Sie können mit Hilfe des doppelten Schrägstrichs // Kommentare mitten in einer Befehlszeile beginnen lassen. Nutzen Sie diese Technik, um wichtige Variablen am Ort ihrer Deklaration zu beschreiben. Beschreiben Sie nicht jede Hilfs- oder Zählvariable, das führt zu unübersichtlichem Code. Denken Sie auch daran, dass Variablenamen selbsterklärend sein sollten und zusätzliche Informationen nicht zum Regelfall werden.

Am Beginn des Skripts selbst sollte ebenfalls eine kurze Beschreibung der Funktionalität stehen. Auf komplexe Algorithmen kann hingewie-

sen werden. Es ist bei größeren Projekten sinnvoll, den Namen des Autors, das Datum der Freigabe und evtl. den Werdegang des Projekts (Versionsfolge) zu beschreiben. Sparen Sie nicht unnötig Bytes ein. Es spielt heute keine Rolle, ob Ihre Quelltexte 5 KByte oder 50 KByte groß sind. Diese Kommentare werden über das Internet nicht übertragen, sodass es für die Bandbreite keine Bedeutung hat.

Formatierung und Strukturierung

Um Programmstrukturen lesen zu können, hat sich bei fast allen Programmiersprachen die Strukturierung des Codes durch Einrückungen etabliert. Wenn Sie einen guten Editor haben, können Sie Einrückungen mit der TAB-Taste vornehmen. Sinnvoll ist die Einstellung auf vier bis acht Leerzeichen für einen TAB-Schritt. Damit der Text nicht zu weit ausfасert, empfehle ich vier Leerzeichen als Einrückgröße. Wenn Ihr Editor diese Einstellmöglichkeit nicht bietet, verwenden Sie Leerzeichen. Dabei steht immer der Beginn und das Ende einer Struktur (zum Beispiel `if {...}` oder `switch {...}`) in einer (vertikalen) Ebene. Wenn Ihr Editor mit proportionalen Schriften arbeiten kann (Times, Arial), deaktivieren Sie diese Option. Verwenden Sie immer nichtproportionale Schriften (Courier). Das erleichtert das Erkennen der Struktur.

Der folgende fiktive Quelltext zeigt, wie eine gute Codestrukturierung und Kommentierung aussehen sollte. **Codestrukturierung**

```
/******  
Funktion:  Ermittelt den ersten Namen im userList-Array  
Eingabe:  $strUserList: Liste der zu durchsuchenden Namen  
          $strTargetUser: Der zu suchende Name  
Ausgabe:  Der Index des ersten gefundenen Namens  
          im userList Array  
          Wird nichts gefunden, wird -1 zurückgegeben  
*****/  
function intFindUser ($strUserList, $strTargetUser)  
    $bInFound = false;           // True, wenn gefunden  
    $intFindUser = -1;  
    $i = 0                       // Initialisiere Zähler  
    while ($i <= count($strUserList) && !$bInFound)  
    {  
        if ($strUserList[$i] == $strTargetUser)  
        {  
            $bInFound = True;     // Gefunden, TRUE setzen  
            return $i;           // Rückgabewert = Zähler  
        }  
        $i++;                   // Nächster Index  
    }  
}
```

Listing 3.101:
Fiktiver Quellcode in
»Idealformatierung«

Die Kommentierung der einzelnen Zeilen kann auch sparsamer ausfallen. Beachten Sie aber die Variablennamen, die sich größtenteils selbst erklären (*strUserList()* → Zeichenkettenarray einer Namensliste). Beachten Sie auch, dass einfache Zähl- oder Hilfsvariablen nicht mit beschreibenden Namen bedacht werden. Die extensive Anwendung in Programmen würde schnell zu einer Einschränkung der Lesbarkeit führen. Für Zählvariablen hat sich die Buchstabengruppe *i, j, k* usw. eingebürgert, für die Zwischenspeicherung von numerischen Werten sind *a, b, c* oder *x, y, z* usw. üblich. Ansonsten kann die Benennung vom Kontext abhängig gemacht werden, das heißt, der umgebende Befehl sollte in seiner kompletten Form mit Variablennamen gut lesbar sein.

Wie im Beispiel zu sehen ist, wird konsequent englisch geschrieben und (nur für den deutschen Markt) deutsch kommentiert.

Lesbarkeit

Die saubere Trennung von HTML-Code und Skript ist für die Lesbarkeit des Quelltextes ebenfalls wichtig. Bei komplexen Anwendungen entsteht schnell ein Durcheinander von Befehlen, die auch geübten Programmierern jede Chance nehmen, eine Funktion zu erkennen oder einen Fehler zu finden.

Haben Sie wenige Skriptbefehle innerhalb des HTML-Codes, sollten Sie jeden Befehl auf eine Zeile setzen und jeweils einzeln mit den Zeichen `<? ?>` umgeben:

```
<BODY>
<H1>Codeanordnung</H1>
<?php if ($name=="admin") { ?>
Hier die <B>internen</B> Codes.
<?php } else { ?>
Hier die &ouml;ffentlichen Codes.
<?php } ?>
</BODY>
```

Sind dagegen längere Passagen in PHP und andere wiederum in HTML geschrieben, dann fassen Sie die Skriptblöcke zusammen. In diesem Fall wird die Einleitung des Skriptblocks `<?php` oberhalb auf eine einzelne Zeile gesetzt, die Ausleitung `?>` ebenso auf eine einzelne Zeile am Ende.

```
<BODY>
<H1>Codeanordnung</H1>
<?php
if ($name=="admin") {
    echo "Hier die <B>internen</B> Codes.";
} else {
    echo "Hier die &ouml;ffentlichen Codes.";
}
?>
</BODY>
```

Generell sollten Sie `echo` nicht benutzen, um viel HTML-Code auszugeben. Setzen Sie besser die umgebenden Skriptbefehle in einzelne `<?php...?>`-Codes.

Wenn es sich nicht vermeiden lässt oder die Lesbarkeit dadurch nicht besser wird, kann es sinnvoll sein, HTML-Code in Konstanten zu packen. Das kommt vor allem bei Tabellenstrukturen vor. Das folgende Beispiel macht davon Gebrauch und definiert eine Art »Super-Tag«:

Achten Sie auf die Lesbarkeit des Code!

```
<?php
define("TABBEGIN","<TR><TD><FONT FACE='Arial'>");
define("TABEND","</TD></TR></FONT>");
?>
<TABLE>
<?php for($i=0;$i<$end;$i++) { ?>
    <?php echo TABBEGIN; ?>
    Laufwerksbuchstabe: <?php echo $strLwName; ?>
<?php echo TABEND; } ?>
</TABLE>
```

Schwer lesbar sind auch lange Variableninhalte. Vor allem im Datenbankbereich kommen komplexe SQL-Abfragen vor. Der folgende Befehl erstreckt sich über mehrere Zeilen, wenn er direkt angewandt wird:

```
<?php $query = "SELECT name, partner, firma, vorwahl, telefon FROM
namen, firmen, kommunikation WHERE kommunikation.vorwahl BETWEEN
'08' AND '08999' GROUP BY NAME" ?>
```

Besser sieht es aus, wenn die Struktur des SQL-Kommandos klar erkennbar ist:

```
<?php
query = "SELECT name, partner, firma, vorwahl, telefon";
query .= " FROM namen, firmen, kommunikation";
query .= " WHERE kommunikation.vorwahl";
query .= " BETWEEN '08' AND '08999' GROUP BY NAME";
?>
```

Achten Sie in solchen Fällen auf die korrekte Anordnung der Leerzeichen (hier in der zweiten bis vierten Variablenzuweisung). Wichtig sind auch einfache und doppelte Anführungszeichen.

Abfrage von Zuständen

Wenn Sie mitten im Code wissen müssen, was der Inhalt einer Variablen oder der Zustand einer Funktion ist, fügen Sie `echo`-Befehle ein. Kennzeichnen Sie diese Stellen, so dass die Ausgaben später leicht entfernt werden können:

```
echo $test;    // debug
```

In sehr großen Projekten bietet es sich an, diese Fehlerausgaben zentral ein- oder ausschalten zu können.

```
if ($debug) { echo $test; }
```

Sie können dann die Ausgaben mit

```
$debug=true;
```

einschalten und mit

```
$debug=false;
```

ausschalten. Sinnvoll ist auch die Nutzung der bereits vorgestellten Funktionen:

- `empty`
Prüft, ob eine Variable leer ist.
- `isset`
Ermittelt, ob eine Variable bereits initialisiert wurde.
- `is_array`
Stellt fest, ob die Variable ein Array ist (kann auch leer sein).
- `is_double`, `is_float`, `is_real`
Diese Funktionen geben TRUE zurück, wenn die Variable einen Gleitkommawert enthält.
- `is_int`, `is_integer`, `is_long`
Diese Funktionen geben TRUE zurück, wenn die Variable einen Ganzzahlwert enthält.
- `is_object`
Wird TRUE, wenn es sich um ein Objekt handelt.
- `is_string`
Wird TRUE, wenn es sich um eine Zeichenkette handelt.
- `gettype`
Ermittelt den Datentyp, der in der Form »string« oder »object« usw. ausgegeben wird.

Mehr Informationen darüber finden Sie in ➡ Abschnitt 3.2.4 *Datentypen* ab Seite 161.



Hinweis

Leider ist der interne Debugger nie fertig entwickelt worden, sodass Sie vieles von Hand machen müssen. Theoretisch kann man Nachrichten des Parsers an einem Port abfangen und daraus Debug-Funktionen entwickeln. Offensichtlich besteht bei Zend wenig Interes-

se daran, den Debugger innerhalb des Standardlieferumfangs von PHP zu entwickeln, denn es gibt eine eigene – kostenpflichtige – IDE, die angeboten wird.

Sie können neben Variablen auch die Existenz von Funktionen abfragen. Dies mag unsinnig erscheinen, denn eigentlich sollten Sie als Programmierer wissen, welche Funktionen bereitstehen. Einsatzmöglichkeiten bestehen aber in der Testphase, zur Kontrolle eingebundener Funktionen und externer Bibliotheken. Möglicherweise bestehen auch Plattform- und Versionsunterschiede, Ihr Code soll aber resistent gegen Veränderungen der Umgebung sein. Dann sollten sie Funktionen testen, die in älteren Versionen nicht vorhanden sind.

```
<?php
function myfunction() {
    // Pseudofunktion
}
if (function_exists('myfunction')) {
    echo "Funktion 'myfunction' ist definiert";
};
?>
```

function_exists()

*Listing 3.102:
Beispiel testfunction:
Funktionen auf
Existenz testen*

Die Anwendung dieser Funktion ist oft im Zusammenhang mit Skripten zu sehen, die noch für PHP 3 entwickelt werden. Falls das Skript auf PHP 4 zum Einsatz kommt, sollte die eingebaute Funktion genutzt werden. Läuft das Skript dagegen mit PHP 3, gibt die Funktion `function_exists` `FALSE` zurück. Innerhalb des `if`-Zweiges wird dann eine Ersatzfunktion definiert.

3.5.6 Hilfe bei der Fehlersuche

Der einfachste Weg zur Fehlersuche ist das systematische Herauskommentieren von Befehlen, die als mögliche Fehlerursache identifiziert wurden oder in Fehlermeldungen benannt sind.

Der Begriff »Herauskommentieren« wird in der Programmierpraxis öfter verwendet. Damit ist gemeint, dass ein Befehl zur Eingrenzung von Fehlerursachen mit einem davor gesetzten Kommentarzeichen deaktiviert wird (in PHP mit dem doppelten Schrägstrich `//`). Wenn Sie den Fehler später gefunden haben, aktivieren Sie den Befehl wieder, indem Sie das Kommentarzeichen löschen.



Um zu erkennen, wie PHP Variablen zuordnet, können Sie die Funktion `var_dump` nutzen. Geben Sie als Argument den Namen einer Variablen an:

var_dump()
print_r()

```
<?php
$test = "abc";
var_dump($test);
echo '<br>';
```

*Listing 3.103:
Beispiel var_dump:
Variablen ausgeben*

```
$test = array(123, 456, 'xyz');
var_dump($test);
?>
```

Etwas umfangreicher, aber vor allem bei Arrays auch unleserlicher, sind die Ausgaben, die `print_r` erzeugt. Beachten Sie, dass bei der Ausgabe globaler Variablen mit einer Referenz auf sich selbst (wie `$GLOBALS`) die Funktion `print_r` in eine Endlosschleife läuft.

Abbildung 3.26:
Ausgabe von
Variablen für
Testanzeige

```
string(3) "abc"
array(3) (
    [0] =>
        int(123)
    [1] =>
        int(456)
    [2] =>
        string(3) "xyz"
```

Die Funktionen eignen sich auch, um den tatsächlichen Datentyp einer Variablen herauszufinden.

3.5.7 Debugging-Tipps

Komplexe Skripte können nicht immer völlig fehlerfrei ablaufen. Oft sind Zustände in Abhängigkeit von Daten zu berechnen, die von Nutzern eingegeben werden. Eine vernünftige Fehlerbehandlung kann einigen Aufwand bei der Erarbeitung aller erdenklichen Systemzustände sparen.

Fehlerbehandlungsroutinen

Generell kann man sich eine Eigenschaft von PHP zunutze machen, die in der Art der Abarbeitung logischer Ausdrücke begründet liegt. Angenommen, Sie rufen Funktionen auf, die möglicherweise Probleme bereiten:

```
$var = @f_kritisch($parm);
```

Dann könnten Sie ebenso gut die folgende Zeile schreiben, ohne dass sich die Funktionsweise ändert:

```
$var = @f_kritisch($parm) or f_fehler($message, $flag);
```

Wie es funktioniert

Für PHP ist dies ein normaler logischer Ausdruck. Aus Performancegründen werden solche Ausdrücke nur so lange bearbeitet, bis das Ergebnis feststeht. Bei einer `or`-Verknüpfung (logisches ODER) ist das Ergebnis `TRUE`, wenn einer der Ausdrücke `TRUE` ist. Wird also die zu beobachtende Funktion `f_kritisch` korrekt abgearbeitet und gibt damit `TRUE` zurück, wird PHP den Ausdruck nicht weiter untersuchen und im Skript fortfahren.

Gibt die Funktion dagegen `FALSE` zurück, muss PHP auch den zweiten Teil untersuchen. Damit wird die individuelle Fehlerbehandlungsrouti-

tine aufgerufen. Das bei der Funktion *f_kritisch* vorangestellte @ unterdrückt die hausinternen Fehlermeldungen und Warnungen.

Als Fehlerbehandlung bietet sich eine Funktion wie die folgende an (funktionsloses Beispiel):

```
function error_check($message, $flag) {  
    printf("<script language=JavaScript>");  
    printf("alert(%s\n\n%s)", $message, $php_errormsg);  
    printf("</script>");  
    if ($flag) return false;  
    return true;  
}
```



Die Anzeige mit einem JavaScript-Fenster verhindert, dass das Layout der Seite durch die Meldungen zerstört wird. Der Mini-Debugger im nächsten Abschnitt führt die Vorgehensweise konsequent fort.

Ein Mini-Debugger

Programme oder Skripte zu Debuggen kann aufwändiger sein, als den gesamten Code zu kreieren und aufzuschreiben. Oft liegt die Ursache in kleinen Unstimmigkeiten, falscher Interpretation der Befehlsfunktion oder unbedachten Zuständen. Rechnen Sie also von vornherein damit, dass größere Skripte nicht auf Anhieb funktionieren. Es gibt einige relativ simple Wege, Code so zu schreiben, dass die »Entwanzung« schnell vonstatten geht. Unabhängig davon ersetzt auch der beste Debugger nicht ein solides Grundwissen der Webserverprogrammierung. Viele Fehler resultieren aus Verständnisproblemen und solche Fehler sind auch mit Werkzeugen nicht zu finden.

Wenn Sie Variablen zusammensetzen oder einfach nur den Inhalt abfragen möchten, bietet sich folgende Vorgehensweise an: **Zustände anzeigen**

```
if($debug) {  
    echo "Var: ".$var;  
}
```

An zentraler Stelle setzen Sie die Variable

```
$debug = TRUE;
```

Wenn Sie die Fehlersuche beendet haben, wird einfach der Parameter in

```
$debug = FALSE;
```

geändert. Beachten Sie, dass Sie den Zugriff auf die globale Variable *\$debug* innerhalb von Funktionen mit folgendem Code ankündigen müssen:

```
global $debug;
```


Das Debugger-Projekt

In Ermangelung eines professionellen Debuggers können Sie nur auf handgestrickte Lösungen zurückgreifen. Die haben dafür den Vorteil, speziell auf die Bedürfnisse Ihrer Projekte zugeschnitten werden zu können. Auf der CD finden Sie unter /ANWENDUNGEN/DEBUGGER einen praktisch nutzbaren Ansatz. Zur Anzeigesteuerung wird intensiv JavaScript genutzt.

Kern ist das Skript `DEBUG.INC.PHP4`, das mit folgender Anweisung in jede Datei eingeschlossen wird:

```
<? include("debug.inc.php4"); ?>
```

Im Skript wird ein weiteres Fenster geöffnet und mit den Daten des aufrufenden PHP-Skripts gefüllt.



```
<?php
/* Standard: Debugging on */
$__dbg_trace = 1;
```

Die Variable `$__dbg_trace` schaltet die Anzeige der Variablen ein oder aus.

Der Debugger kann neben den individuell ausgewählten Variablen auch die Standardvariablen anzeigen. Dazu gehören:

- `$GLOBALS`

Das Array aller globalen Variablen. Da die Variablen zur Steuerung des Debuggers ebenfalls global sind, werden sie mit einer if-Anweisung ausgeblendet.

- `$HTTP_POST_VARS`

In diesem Array stehen alle per HTTP-Post (aus Formularen heraus) übermittelten Werte (das Thema wird in Kapitel 4 angesprochen).

- `$HTTP_GET_VARS`

Die in der URL übertragenen Werte, ebenso Thema in Kapitel 4.

- `$HTTP_COOKIE_VARS`

Cookies sind für jeden Webserver-Programmierer ein wichtiges Thema.

*Listing 3.104:
Kernfunktionen des
Debuggers im Skript
debug.inc.php4*

```
<?php
$dbgglob = "<br><b>Globale Variablen:</b><code><br>";
if (is_array($GLOBALS)) {
    while (list($key, $val) = each($GLOBALS)) {
        if ($key!="dbgglob" & $key!="key" & $key!="val"
            & $key!="__dbg_trace") {
            $dbgglob .= "<nobr>".$key." =>
```

```

        <font color=green>" . $val .
        "</font></nobr><br>";
    }
}
} else {
    $dbgglob = "<font color=green>Keine ...</font><br>";
}
$dbgglob .= "</code>";
// Formularvariablen
$dbgpost = "<br><b>Variablen per POST:</b><code><br>";
if (is_array($HTTP_POST_VARS)) {
    while (list($key, $val) = each($HTTP_POST_VARS)) {
        $dbgpost .= $key." =>
            <font color=green>" . $val .
            "</font><br>";
    }
} else {
    $dbgpost .= "<font color=green>Keine ...</font><br>";
}
$dbgpost .= "</code>";
// GET-Variablen (QueryString)
$dbgget = "<br><b>Variablen per GET:</b><code><br>";
if (count($HTTP_GET_VARS)>0) {
    while (list($key, $val) = each($HTTP_GET_VARS)) {
        $dbgget .= $key." => <font color=green>".$val."</font><br>";
    }
} else {
    $dbgget .= "<font color=green>Keine ...</font><br>";
}
$dbgget .= "</code>";

// Cookies
$dbgcook = "<br><b>Cookies:</b><code><br>";
if (is_array($HTTP_COOKIE_VARS)) {
    while (list($key, $val) = each($HTTP_COOKIE_VARS)) {
        $dbgcook .= $key." => <font color=green>".$val."</font><br>";
    }
} else {
    $dbgcook .= "<font color=green>Keine ...</font><br>";
}
$dbgcook .= "</code>";

```

Die Fehlermeldungen in PHP-Skripten werden Sie möglicherweise unterdrücken. Die Anzeige im Debugger hilft die Warnungen auch bei laufendem Skript zu erkennen:

```

$dbgerr = "<br><b>Fehlermeldungen:</b><br>";
if (empty($php_errormsg)) {
    $dbgerr .= "<font color=green><code>Keine
        ...</code></font><br>";
} else {
    $dbgerr .= "<font

```

```
color=red><code>".$php_errormsg."</code></font><br>";
}
?>
```

Der Effekt eines gesonderten Fensters beruht auf einer JavaScript-Routine. Der Befehl `window.open` öffnet ein neues Fenster mit bestimmten Eigenschaften:

JavaScript

```
<script language="JavaScript">
debwin = window.open("debug.htm", "PHPDebugger",
"alwaysRaised=yes, directories=no, hotkeys=no,
location=no, resizable=no, screenX=0, screenY=0, status=no,
titlebar=no, toolbar=no, width=300, height=500");
```

Von PHP zu JavaScript: Mit den folgenden Zeilen werden die in PHP berechneten Werte (Inhalte der Variablen-Arrays) nach JavaScript übertragen. Die Methode ist insofern wichtig, als hiermit eine Übertragung von Variablen eines Serverskripts direkt in Variablen eines Browser-Skripts stattfindet:

```
debwin.dbgpost = "<?php echo $dbgpost; ?>";
debwin.dbgget = "<?php echo $dbgget; ?>";
debwin.dbgglob = "<?php echo $dbgglob; ?>";
debwin.dbgcook = "<?php echo $dbgcook; ?>";
debwin.dbgerr = "<?php echo $dbgerr; ?>";
```

Die HTML-Seite wird komplett aus dem Skript heraus erzeugt. Bedenken Sie, dass hier das originale, zu debuggende PHP-Skript abläuft. Die JavaScript-Befehle senden die ermittelten Inhalte an das neue, zusätzlich geöffnete Fenster:

```
debwin.control.document.write("<HTML><HEAD></HEAD>
                                <BODY bgcolor=#fafad2>\n");
debwin.control.document.write("<SPAN
                                onclick='parent.debug.document.write(parent.dbgpost)'>
                                <U> [POST] </U></SPAN>\n");
debwin.control.document.write("<SPAN
                                onclick='parent.debug.document.write(parent.dbgget)'>
                                <U> [GET] </U></SPAN>\n");
debwin.control.document.write("<SPAN
                                onclick='parent.debug.document.write(parent.dbgglob)'>
                                <U> [GLOB] </U></SPAN>\n");
debwin.control.document.write("<SPAN
                                onclick='parent.debug.document.write(parent.dbgcook)'>
                                <U> [COOK] </U></SPAN>\n");
debwin.control.document.write("<SPAN
                                onclick='parent.debug.document.write(parent.dbgerr)'>
                                <U> [ERR] </U></SPAN>\n");
debwin.control.document.write("</BODY></HTML>\n");
debwin.debug.document.write("<br><b><font size=-2>
                                [Zeile] (Typ) Zustand/Inhalt {Name/Bemerkung}
                                </font></b><p>");
</script>
```



```
<frame src="" noresize name="debug">
</frameset>
</HTML>
```

Die beiden Frames bekommen hier ihren Namen mit folgenden Tags:

Wie es funktioniert

```
<frame src="" noresize name="control">
<frame src="" noresize name="debug">
```

Damit ist die Zuordnung im JavaScript-Code möglich.

Der Debugger ist sicher kein echter Ersatz für professionelle Werkzeuge, viele Funktionen fehlen oder sind mit den hier gezeigten Mitteln nicht realisierbar. Dennoch ist er ein erster Ansatz für eigene Versuche. Die Anwendung ist einfach. Wenn Sie eine bestimmte Stelle in Ihrem Code überwachen möchten, bauen Sie folgende Zeile ein:

Nutzung des Debuggers

```
@debug(__LINE__, $variable_zu_test, "Name/Kommentar");
```

Wenn das Skript diese Stelle erreicht, wird die Zeilennummer, der Inhalt der Variablen (auch wenn es ein Array ist), der Datentyp und der Kommentar dazu in einem extra Fenster ausgegeben.

Mit der folgenden Angabe werden die Ausgaben unterdrückt:

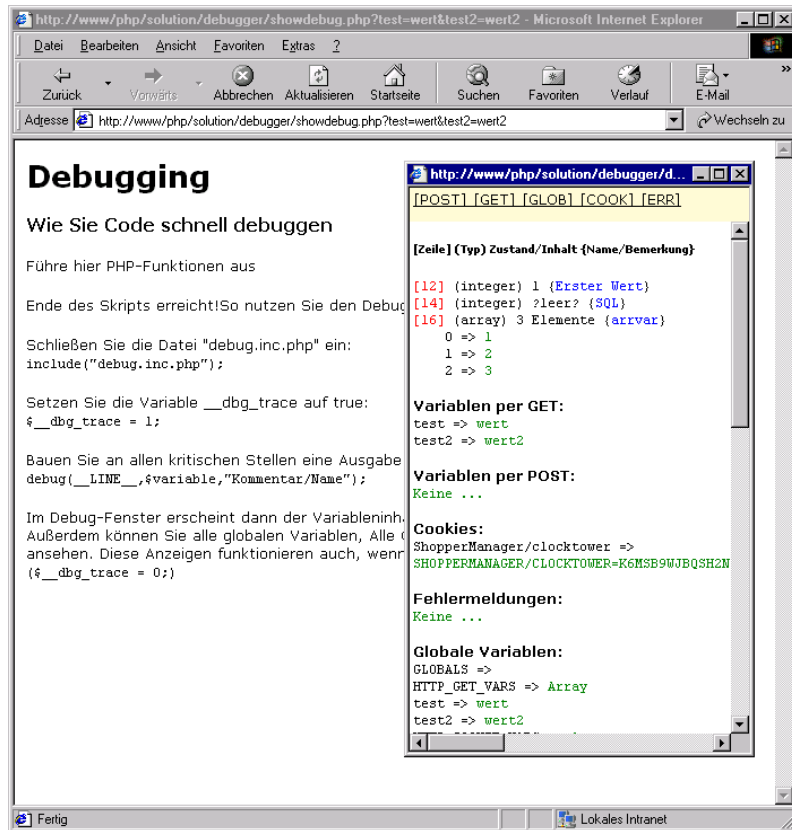
```
$ _dbg_trace = FALSE;
```

Entsprechend wird mit nachfolgendem Code die Anzeige wieder eingeschaltet:

```
$ _dbg_trace = TRUE;
```

Der Standardwert ist die eingeschaltete Anzeige. Sind alle Skripte getestet und laufen einwandfrei, wird einfach die `include`-Anweisung entfernt.

Abbildung 3.27:
Der Mini-Debugger
in Aktion: Das
kleinere Fenster zeigt
die Ausgaben bei
laufendem Skript



3.6 Hilfsfunktionen

Die hier beschriebenen Hilfsfunktionen dienen der Lösung von seltener auftretenden Problemen. Da sie oft elegante Skripte ermöglichen, sollten sich auch Anfänger damit beschäftigen.

Übersicht

Dieser Abschnitt beschäftigt sich mit Funktionen, die viele Programmierprobleme einfach und elegant lösen. Die Anwendung ist dennoch nicht Standard, die Funktionen sind weniger bekannt und haben oft keine Entsprechung in anderen Sprachen. Hier eine Übersicht:

- extract

Extrahiert ein Array und legt die Elemente in einzelnen Variablen ab, die eigens zu diesem Zweck erzeugt werden.

- `eval`

Hiermit wird PHP-Code ausgeführt. Damit lässt sich dynamisch generierter Code zum Ablauf bringen.

- `get_html_translation_table`

Diese Funktion gibt die Tabelle der HTML-Entitäten und ihrer Entsprechung in ASCII als assoziatives Array zurück.

Variablen aus einem Array erzeugen: `extract`

Die Funktion `extract` liest ein assoziatives Array ein und erzeugt für jeden Wert eine Variable. Um die Variablen von anderen Variablen im Skript unterscheiden zu können, kann ein Präfix angegeben werden. **`extract()`**

```
extract (array, type, prefix);
```

Syntax

Mit dem Parameter `type` wird angegeben, wie die Umwandlung erfolgen soll. Sie können auf folgende Konstanten zurückgreifen:

- (0) `EXTR_OVERWRITE`

Wenn der erzeugte Name mit einer vorhandenen Variablen kollidiert, wird die vorhandene Variable überschrieben. Diese Option ist der Standardwert.

- (1) `EXTR_SKIP`

Wenn der erzeugte Name mit einer vorhandenen Variablen kollidiert, wird die vorhandene Variable nicht überschrieben.

- (2) `EXTR_PREFIX_SAME`

Wenn der erzeugte Name mit einer vorhandenen Variablen kollidiert, wird der Präfix auf die Variable angewendet, die von der Kollision betroffen ist.

- (3) `EXTR_PREFIX_ALL`

Alle Variablen werden zur Vermeidung von Kollisionen mit dem Präfix versorgt.

Jede Konstante steht für einen Integer-Wert (in Klammern angegeben). Es ist zu empfehlen, die Option `EXTR_PREFIX_ALL` zu nutzen.

```
<?php
$array = array("as" => "Active Server Pages",
               "pl" => "Perl",
               "ph" => "PHP");
extract($array, EXTR_PREFIX_ALL, "strScript");
echo $strScript_as."<BR>";
echo $strScript_pl."<BR>";
echo $strScript_ph."<BR>";
?>
```

*Listing 3.107:
Beispiel `misc_extract`:
Anwendung der
Funktion `extract`*

Wenn der Präfix verwendet wird, besteht der komplette Variablenname aus dem Präfix, einem Unterstrich und dem Schlüssel des assoziativen Arrays.

Wenn der Schlüssel kein gültiger Variablenname ist, wird der Vorgang für diesen Eintrag nicht ausgeführt.

PHP-Code implizit ausführen: eval

eval()

Der Befehl eval führt den angegebenen Code sofort aus. Der Inhalt von Variablen wird aus der umgebenden Ebene übernommen, globale Variablen müssen also auch hier bekannt gemacht werden, wenn der Aufruf innerhalb einer Funktion erfolgt.

Listing 3.108:
Beispiel *misc_eval*:
Anwendung von
eval

```
<?php
$test = "Ausgabe mit eval";
$output = "echo";
eval("$output \$test;");
?>
```

Beachten Sie, dass auf Variablen innerhalb der zu überprüfenden Zeichenketten nur zuverlässig zugegriffen werden kann, wenn vor dem \$-Zeichen ein Backslash steht.

HTML-Sonderzeichen bearbeiten: get_html_translation_table

Auf die Möglichkeit, die Sonderzeichen in HTML zu erkennen und umzuwandeln, wurde bereits im ➡ Abschnitt *HTML-spezifische Funktionen* ab Seite 199 eingegangen. Die Funktionen htmlspecialchars und htmlentities basieren auf einer internen Tabelle, der sogenannten HTML-Translationtable. Jede Funktion verwendet dabei einen eigenen Satz von Vergleichswerten, die wie ein assoziatives Array angeordnet sind.

strtr()

Umsetzungen von Werten nimmt auch die Funktion strtr vor. Normalerweise wird ein Array erwartet, das die umzuwandelnden Werte enthält. Wenn man nun die HTML-Tabelle mit der Funktion get_html_translation_table ermittelt, kann das resultierende Array erweitert und auf strtr angewendet werden. So wird eine Erweiterung der internen Umwandlungsmöglichkeiten vereinfacht.

Interessant ist eine andere Anwendung, die oft benötigt wird. Die Funktionen htmlspecialchars und htmlentities haben keine Umkehrfunktion. Sie können dies aber simulieren, wenn Sie die Richtung der Vergleichstabelle umdrehen:

Listing 3.109:
misc_htmltrans: Die
HTML-Übersetzung
umdrehen

```
<?php
function ret_entities($html) {
    $trans = get_html_translation_table(HTML_ENTITIES);
    $trans = array_flip($trans);
    return strtr($html, $trans);
}
```

```
}  
  
$coded = "Ein K&uuml;l&szlig;chen f&uuml;r Haide...";  
echo "<br>Original: " . htmlspecialchars($coded);  
echo "<br>Ursprung: " . ret_entities($coded);  
?>
```

Als »Drehfunktion« kommt `array_flip` zum Einsatz. Die Ausgabe sollte dann etwa folgendermaßen aussehen:

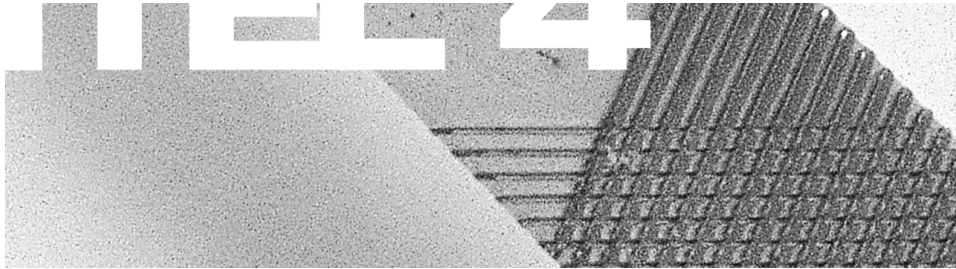
```
Original: Ein K&uuml;l&szlig;chen f&uuml;r Haide...  
Ursprung: Ein Kü&szl;chen für Haide...
```

Prüfen Sie im Quelltext der Seite, dass hier tatsächlich keine HTML-Symbole verwendet wurden, sondern die Zeichen des originalen Zeichensatzes.

Abbildung 3.28:
Ausgabe des Skripts
aus Listing 3.109

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

4



Interaktive Webseiten

Mit PHP dynamisch erstellte Webseiten machen erst Sinn, wenn der Nutzer damit interagieren kann. In diesem Kapitel lernen Sie die entsprechenden Techniken im Detail kennen.

Formulare auswerten (Seite 301)

Daten per URL weiterreichen (Seite 318)

Cookies (Seite 324)

Verbindungssteuerung (Seite 333)

Sessionverwaltung (Seite 336)

Zugriff auf das Dateisystem (Seite 350)

Verbindungen zu Servern im Internet (Seite 378)

Sicherheit (Seite 400)

Bilderzeugung (Seite 411)

4.1 Formulare auswerten

Formulare sind der Schlüssel zu interaktiven Webseiten. Der Nutzer wird in die Lage versetzt, Daten einzugeben, und der Server kann auf diese Daten in vielfältiger Weise reagieren. Die Darstellung der Formulare erfolgt mit Hilfe dafür vorhandener HTML-Tags, den Formular-Elementen. Die Übertragung der Daten zum Server übernimmt HTTP mit den Methoden POST oder GET.

4.1.1 GET und POST

Das Verständnis für die Art der Datenübertragung macht in einigen Fällen die Programmierung einfacher. Unabhängig davon ist es für Entwickler professioneller Sites immer sinnvoll, über etwas Hintergrundwissen zu verfügen. Die Darstellung der Übertragungsmethoden für Formulare entstammt der Protokollbeschreibung für HTTP, veröffentlicht in der RFC 2068. Der folgende Text ist eine lockere, leicht gekürzte Übersetzung der Abschnitte 9.3 *GET* und 9.5 *POST* der RFC 2068.



RFC 2068

POST

Die POST-Methode wird verwendet, um dem Server mitzuteilen, dass die gesamte Einheit der Anforderung weitere Daten im Körper der Nachricht enthält. POST erlaubt damit unter anderem folgende Funktionen:

Definition: POST

- Mitteilung existierender Ressourcen
- Übertragung von Nachrichten auf eine Nachrichtengruppe, Mailingliste oder ähnliche Gruppen
- Übertragung eines Datenblocks (dazu gehören auch die Inhalte eines Formulars)

Die aktuell durch POST ausgelöste Aktion wird durch den Server bestimmt und ist abhängig von der Request-URI. Der gesendete Datenblock ist Bestandteil dieser URI, ähnlich wie eine Datei Bestandteil eines Verzeichnisses oder eine Nachricht Bestandteil einer Nachrichtengruppe ist. Der durch POST ausgelöste Prozess muss nicht direkt an eine Ressource gerichtet sein, die durch den URI adressiert wird. In diesem Fall wird entweder 200 (OK) oder 204 (No Content) als Antwortstatus gesendet. Wird eine Ressource auf dem Server erzeugt, wird 201 (Created) gesendet, im Körper der Nachricht steht eine Beschreibung des Status der Anforderung und ein Verweis auf die neue Ressource einschließlich des neuen Kopfes. Alternativ kann die Antwort 303 (See Other) zur Umleitung des Clients gesendet werden.

Das ist der theoretische Aspekt. Praktisch überträgt dieser Mechanismus Formulardaten vom Browser zum Server, mehr muss der Programmierer am Anfang nicht wissen.

GET

Definition: GET

Mit der GET-Methode werden Informationen – welcher Art auch immer – durch die Ergebnis-URI identifiziert. Wenn der URI auf einem Prozess basiert, der Daten zurückgibt, besteht er aus den produzierten Daten. Die komplette URI besteht aus der URL, einem Fragezeichen als Trennzeichen und den Daten. Es ist möglich, eine GET-Anforderung mit Bedingungen zu senden (If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match, If-Range). In diesen Fällen werden die Daten nur übertragen, wenn die Bedingungen zutreffend sind.

Anwendung

Die theoretischen Formulierungen in der RFC 2068 sind wenig geeignet, daraus ein Skript entwickeln zu können. Auch die gesamte RFC, insgesamt über 160 Seiten, bringt wenig. Dennoch ist die Unterscheidung der Methoden wichtig, denn sowohl HTML (mit dem <FORM>-Tag) als auch PHP mit den Anfragefunktionen reflektieren die hier verwendeten Begriffe.

Grundsätzlich sind zum Verständnis der Methoden die Grundlagen aus ➡ Abschnitt 2.2 *Höhere Netzwerkprotokolle* (siehe Seite 72) notwendig. Wenn Ihnen der ungefähre Vorgang beim HTTP-Protokoll klar ist, werden auch die hier verwendeten Begriffe transparenter. HTTP-Nachrichten bestehen aus dem anfordernden URI, dem Kommando und der eigentlichen Nachricht. Die Nachricht besteht aus Kopf (Header) und Körper (Body).

Für den hier vorgestellten Zweck ist nur interessant, dass die Methode POST die Daten im Körper verpackt und so als Teil der Nachricht überträgt. Hier ist der Platz nicht beschränkt. GET nutzt dagegen die URI zur Übertragung, hängt die Daten quasi an die URL mit an. Die Länge der URL ist aber Beschränkungen von Seiten der Browser unterworfen, akzeptable Größen liegen bei 2 KByte. Dies ist schnell erreicht, denn neben den Daten zählt auch jedes Trennzeichen und jeder Variablen- oder Feldname.

Mit POST umgehen Sie solche Beschränkungen. In hochperformanten Umgebungen kann als einziger Nachteil der größere Zeitaufwand angesehen werden, da der Server zusätzlich zum Kopf der Nachricht auch den gesamten Körper erkennen und weiterreichen muss. In der Praxis dürfte der Unterschied irrelevant sein.

4.1.2 Daten aus einem Formular ermitteln

Formulare werden enorm leistungsfähig, wenn es gelingt, die Daten in einem Skript weiterzuverarbeiten. Wie dies erfolgt, finden Sie in diesem Abschnitt.

Das Formular

HTML-Formulare sind der einzige praktikable Weg, auf dem Daten vom Browser zum Server und damit zu Ihrem Skript gelangen können. Ein Formular besteht aus dem `<FORM>`-Tag mit verschiedenen Parametern und den Feld-Elementen, HTML-Tags zur Darstellung von Eingabefeldern.

HTML-Formulare

HTML-Eingabefelder können nur innerhalb des `<FORM>`-Containers existieren. Außerhalb werden Sie je nach Browser nicht oder falsch dargestellt, funktionslos sind sie dort auf jeden Fall.

Eine einfache Anwendung kennen Sie vielleicht bereits:

```
<HTML>
<HEAD><TITLE>Mail-Formular</TITLE></HEAD>
<BODY>
<FORM action="mailto:joerg@krause.net">
  Nachricht: <INPUT type="text" name="Message" size="50">
  E-Mail: <INPUT type="text" name="eMailaddress" size="50">
</FORM>
</BODY>
</HTML>
```

*Listing 4.1:
Einfaches Formular
in HTML, noch ohne
PHP*

Das einfache Formular aus Listing 4.1 sendet eine E-Mail. Der gesamte Prozess läuft allerdings clientseitig ab. Voraussetzung ist, dass neben dem Browser auch ein E-Mail-Client installiert ist, der vom Browser auch gesteuert werden kann. Vor allem im Intranet ist dies oft nicht der Fall.

In solchen Fällen ist ein Skript auf dem Server notwendig, das unabhängig vom Browser und dessen Installationsumgebung E-Mails versenden kann. Praktisch ist dazu nur eine Zeile auszutauschen:

```
<FORM action="sendmail.php" method="post">
```

Zwei Dinge sind hier anders: zum einen wird als auszuführender Befehl für den `<FORM>`-Tag nicht mehr eine lokale Anweisung angegeben, sondern ein Skript auf dem Server. Zum anderen wird explizit auf die zu verwendende Datenübertragungsmethode gezeigt: POST.

Browserseitig können Formulare ausgewertet und der Inhalt der Felder beeinflusst werden, indem eine Skriptsprache eingesetzt wird, die der Browser versteht, beispielsweise JavaScript.



Im Beispiel wurde bereits gezeigt, wie ein Text-Feld erzeugt wird. In Tabelle 4.1 finden Sie eine Übersicht über alle in HTML erlaubten Feldelemente.

Tabelle 4.1:
HTML-
Formularelemente

Elementtyp	Beschreibung	Attribute
text	Einzeiliges Eingabefeld	size, value, name
checkbox	Kontrollkästchen	value, checked, name
radio	Optionsschaltfläche	value, checked, name
submit	Sendeschalter	value, name
reset	Schalter zum Rücksetzen	value, name
password	Verdecktes Eingabefeld	size, value, name
hidden	Unsichtbares Feld	value, name
button	Schaltfläche	value, name
image	Bild, ersetzt submit	src, name, Bildattribute
file	Eingabefeld und Schalter	name, accept

Die Attribute haben folgende Bedeutung:

- **size**
Länge des Feldes in Standardzeichen. Die genaue Ausdehnung hängt von der Schriftart ab.
- **value**
Vorbelegter Wert (text, hidden) oder Beschriftung (button, reset, submit). Dieser Wert wird auch gesendet (bei Text der vom Nutzer ggf. geänderte Inhalt).
- **name**
Name, nach dem das Element im Skript identifiziert werden kann.
- **checked**
Das Element ist aktiv (radio und checkbox).
- **src**
Ort, wo sich das Bild befindet.

Neben diesen einfachen Elementen gibt es noch zwei Elemente, die als Container existieren:

- **<SELECT>**
Dieses Element stellt eine Werteliste in Form eines Drop-Down-Menüs dar. Jedes Element wird durch ein weiteres Tag, **<OPTION>**, eingeleitet.

- <TEXTAREA>

Mit diesem Element werden mehrzeilige Eingabefelder für Text realisiert.

Allzu umfangreich sind die Möglichkeiten zur Formulargestaltung also nicht – gemessen an dem, was aus der Windows- oder X-Programmierung bekannt ist.

Wenn Sie im Umgang mit diesen Elementen unsicher sind, konsultieren Sie eine HTML-Dokumentation. Die hier vorgestellten Anwendungen arbeiten mit Elementen, wie sie bereits unter HTML 3.2 definiert wurden.



Auswertung

Die Auswertung und Übergabe der Formulardaten in PHP ist ausgesprochen einfach. PHP erkennt selbstständig angehängte Daten, egal ob mit GET in der URI oder mit POST im Körper der Nachricht. Jedes Formularelement ist durch das Attribut name gekennzeichnet. Es erscheint in PHP als normale Variable mit dem Namen des Elements, gefüllt mit den eingegebenen Werten. Durch Bilder erzeugte Schaltflächen übermitteln in den Variablen `$name_x` und `$name_y` die Position des Mauszeigers, relativ zur linken oberen Ecke des Bildes.

```
<?php
if ($sent & $name) {
    echo <<<FORMANSWER
    Sie haben folgende Nachricht gesendet:<p />
    <b>Ihr Name:</b> $name<br />
    <b>E-Mail:</b> $email<br />
    <b>Bewertung:</b> $rank<br />
    <b>Ihre Nachricht:</b> <br />
    <div style="background-color:#DDDDDD">$message</div>
    <b>Gekauft bei: </b>$purchase
    <form method="post" action="$PHP_SELF">
    <input type="submit" value="Neue Nachricht senden?" />
    </form>
FORMANSWER;
} else {
?>
<h3>Wie hat Ihnen das Buch bisher gefallen?</h3>
<form method="post" action="$PHP_SELF">
<table>
    <tr>
        <td>Mein Name:</td>
        <td><input type="text" name="name" size="50" /></td>
    </tr>
    <tr>
        <td>Meine E-Mail:</td>
        <td><input type="text" name="email" size="50" /></td>
```

Listing 4.2:
form_getval:
Komplexe Formulare
auswerten

```

</tr>
<tr>
  <td>Bewertung:</td>
  <td>
    1<input type="radio" name="rank" value="1" />
    2<input type="radio" name="rank" value="2" checked="checked"/>
    3<input type="radio" name="rank" value="3" />
    4<input type="radio" name="rank" value="4" />
    5<input type="radio" name="rank" value="5" />
    (Schulnoten)
  </td>
</tr>

<tr>
  <td valign="top">Nachricht:</td>
  <td>
    <textarea cols="30" rows="6" name="message">
      Ihre Nachricht...</textarea>
    </td>
</tr>
<tr>
  <td>Wo haben Sie es gekauft?</td>
  <td>
    <select size="1" name="purchase">
      <option value="Fachbuchhandel" />Fachbuchhandel
      <option value="Kaufhaus" />Kaufhaus
      <option value="Online" />Online
      <option value="Verlag" />Verlag
      <option value="Nur geliehen" />Nur geliehen
    </select>
  </td>
</tr>
<tr>
  <td colspan="2">
    <input type="submit" value="Absenden" />
    <input type="reset" />
    <input type="hidden" name="sent" value="1" />
  </td>
</tr>
</table>
</form>
<?php
} /* end if */
?>

```

Wie es funktioniert Das Beispiel mag auf den ersten Blick etwas unübersichtlich erscheinen. Analysieren Sie die Funktion schrittweise:

- Das Skript ruft sich selbst auf, der eigene Name wird der vordefinierten Variablen `$PHP_SELF` entnommen.

- Das Skript ist zweigeteilt, es enthält sowohl das Formular zum Senden als auch das Skript zum Auswerten.
- Die Erkennung, ob es sich um die Darstellungs- oder Auswertephase handelt, wird anhand der Variablen `$sent` getroffen.
- Zusätzlich wird ausgewertet, ob wenigstens der Name ausgefüllt wurde:

```
if ($sent & $name) {
```

- PHP wertet gefüllte Variablen (Formularelemente) als TRUE, ungefüllte oder nicht vorhandene als FALSE.
- In der Auswertephase werden die Daten wiedergegeben. Das Formular kann erneut aufgerufen werden, indem ein neues Formular gesendet wird, das außer der Sendeschaltfläche keine Elemente enthält, die nächste Auswertung verzweigt dann in dem else-Zweig:

```
<form method="post" action="$PHP_SELF">
<input type="submit" value="Neue Nachricht senden?" />
</form>
```

4.1.3 Formularelemente auf Existenz testen

Im Beispiel aus Listing 4.2 wurde bereits von der Tatsache Gebrauch gemacht, dass nicht vorhandene oder nicht ausgefüllte Formularelemente gezielt ausgewertet werden. Es ist daher wichtig zu wissen, wie sich bestimmte Elemente verhalten.

Das Kontrollkästchen (checkbox) überträgt den Wert des Attributes `value`. Wenn `value` nicht angegeben wurde, wird das aktivierte Kontrollkästchen »on« übertragen. Ein nicht aktiviertes Kontrollkästchen überträgt nichts, nicht einmal den Namen selbst.

checkbox

Optionsfelder dienen der Auswahl aus einer fest begrenzten Wertemenge. Gruppen entstehen, indem mehrere dieser Felder den gleichen Namen tragen. Wird keines der Felder ausgewählt, erscheint es nicht als Variable. Wird eines ausgewählt, wird der Wert des Attributes `value` übertragen. Sie können die Angabe eines Wertes erzwingen, indem eine Option mit dem Attribut `checked` gesetzt wird.

radio

Textfelder in allen drei Varianten geben immer wenigstens eine leere Zeichenkette zurück und damit auch den Namen. Bei `text` und `hidden` bestimmt der Inhalt des Attributes `value`, was standardmäßig gesendet wird. Bei `<TEXTAREA>` steht der Standardtext zwischen den Tags.

text
hidden
<textarea>

Auswahlboxen können völlig unselektiert bleiben, dann wird auch der Name nicht übertragen. Sie können aber eine Standardauswahl erzwingen, indem das Attribut `default` gesetzt wird (im `<OPTION>`-Tag).

<select>

submit

Sendeschaltflächen werden nicht übertragen, wenn value nicht ausgefüllt wurde. Die Schaltfläche verwendet normalerweise value als Bezeichner. Wird nichts angegeben, erscheint der Standardtext »Submit« oder »Anfrage absenden«¹³, dieser wird aber nicht gesendet.

4.1.4 Formulare und JavaScript

Formulare sind in ihrer praktischen Anwendung recht eingeschränkt, wenn man sich auf reines HTML und die Auswertung mit PHP beschränkt. Es gibt eine Vielzahl von Problemfällen, die sich ohne zusätzliche Hilfsmittel nur schwer lösen lassen. Das effektivste Hilfsmittel bei der Umsetzung von Formularen ist JavaScript.



JavaScript mag Ihnen in einem PHP-Buch deplatziert erscheinen. In der Praxis sind Sie aber auf die Anwendung angewiesen, um effektiv programmieren zu können. Dem Einsteiger soll dieser Abschnitt deshalb die Richtung zeigen, in der sich eigene Erkundungen sinnvoll anstellen lassen.

JavaScript wird im Browser verarbeitet

Wenn Ihnen der Umgang mit JavaScript noch nicht geläufig ist, sollten Sie sich entsprechende Literatur beschaffen. Schwer ist es nicht, da viele Elemente die gleiche Quelle wie PHP nutzen (C, Java) und deshalb oft ähnliche Sprachkonstrukte entstehen. Der entscheidende Unterschied: JavaScript wird im Browser verarbeitet. Alle modernen Browser verstehen inzwischen JavaScript in einem Umfang, der die Anwendung bedenkenlos ermöglicht. Der Einsatz bietet handfeste Vorteile:

- Einsparung an Netzverkehr
- Elegantere Überprüfung von Formularen
- erweiterte Funktionalität

Nicht alles ist jedoch allein mit JavaScript möglich. Professionelle Anwendungen entstehen durch das optimale Zusammenspiel von HTML, JavaScript und PHP.

JavaScript in der HTML-Seite

<SCRIPT>

JavaScript kann, ebenso wie auf der Serverseite PHP, überall eingebunden werden. Dazu verwenden Sie das <SCRIPT>-Tag:

```
<SCRIPT language="JavaScript">
// Hier steht der JavaScript-Code
</SCRIPT>
```

¹³ Die konkrete Ausschrift hängt von der Sprache des Browsers und dem Browser selbst ab.

Wenn Sie umfangreiche Programme schreiben, die von mehreren Seiten verwendet werden, kann die Auslagerung in eine eigene Datei sinnvoll sein:

```
<SCRIPT language="JavaScript" src="meincode.js">
// Hier steht der JavaScript-Code
</SCRIPT>
```

Funktionen, die auf der Seite häufiger verwendet werden, platzieren Sie am besten im Kopf (zwischen </TITLE> und <BODY>). Kleinere Einzeiler sind oft besser an der Stelle aufgehoben, wo sie auch zum Einsatz gelangen.

JavaScript erweitert eine ganze Reihe von HTML-Elementen um ein Ereignismodell. Die Elemente können nun vor allem auf Mausereignisse reagieren. In Formularen sind folgende Ereignisse wichtig:

- onsubmit

Dieses Ereignis wertet den Klick auf den Absende-Schalter aus und führt den JavaScript-Code aus, der durch das Attribut onsubmit spezifiziert wird.

- onclick

Wertet einen Mausklick aus. Damit werden vor allem einfache Schaltflächen <INPUT type=button> zum Leben erweckt.

JavaScript einbinden

In Listing 4.2 wurde bereits ein komplexes Formular vorgestellt. JavaScript könnte gut genutzt werden, um das Ausfüllen des Formulars zu überwachen. Dazu wird das <FORM>-Tag verändert:

```
<FORM name="sendform" onsubmit="return check_data()">
```

Das sieht nun völlig anders aus als im ersten Beispiel. Zum einen hat das Formular einen Namen bekommen. Dies hat primär nichts mit JavaScript zu tun, erleichtert aber den Umgang mit mehreren Formularen. Von PHP aus können Formularnamen nicht erkannt werden. Zum anderen wird statt des action-Attributs die Aktion nun mit onsubmit ausgelöst, verbleibt also zunächst im Browser. Was passiert nun? Wenn das Formular mit den Schalter »Absenden« vom Nutzer gesendet wird, löst der Browser das JavaScript-Ereignis onsubmit aus. Dort steht der JavaScript-Code: return checkdata(). Die Syntax kennen Sie bereits aus PHP: return gibt einen Funktionswert zurück, checkdata() ruft eine Funktion auf. In diesem Fall kann return nur TRUE oder FALSE zurückgeben, bei TRUE wird das Formular tatsächlich gesendet – aber wohin?

Die Funktion checkdata() muss offensichtlich selbst programmiert werden (wie Sie Funktionen benennen, spielt keine Rolle, solange es sich

*Listing 4.3:
Ausschnitt aus
form_javascript:
Überprüfung eines
Formulars mit
JavaScript (der Rest
des Skripts entspricht
Listing 4.2)*

nicht um reservierte Namen in JavaScript handelt). Im Kopf der Seite wird dazu folgender Code eingefügt:

```
<script language="JavaScript">
function checkdata(){
  var re_email = /^[_a-zA-Z0-9-]+\.[_a-zA-Z0-9-]*
    @([_a-zA-Z0-9-]+\.)+([a-zA-Z]{2,3})$/;
  var email = document.sendform.email.value;
  var name = document.sendform.name.value;
  var checked = true;
  if (name.length<=3)
  {
    alert("Name muss mindestens 3 Zeichen enthalten");
    checked = false;
  }
  if ((re_email.test(email))==false || email.length==0)
  {
    alert("E-Mail-Adresse falsch!");
    checked = false;
  }
  if (checked)
  {
    document.sendform.method = "post";
    document.sendform.action = "<? echo $PHP_SELF ?>";
    document.sendform.submit();
  }
  return checked;
}
</script>
```

Wie es funktioniert

Geprüft wird im Beispiel, ob der Name wenigstens drei Zeichen enthält und die E-Mail-Adresse dem normalen Aufbau genügt. Da dieser Test etwas umfangreicher ist, wird ein regulärer Ausdruck verwendet. Die Entwicklung dieses Ausdrucks kennen Sie bereits aus ➤ Abschnitt 3.2.12 *Reguläre Ausdrücke* ab Seite 217. JavaScript kann mit regulären Ausdrücken ohne Einschränkungen umgehen. Die Syntax entspricht der bei Perl verwendeten. Zuerst wird also das Muster festgelegt:

```
var re_email = /^[_a-zA-Z0-9-]+\.[_a-zA-Z0-9-]*
  @([_a-zA-Z0-9-]+\.)+([a-zA-Z]{2,3})$/;
```

Dann werden die Inhalte der Formularelemente in Variablen überführt:

```
var email = document.sendform.email.value;
var name = document.sendform.name.value;
```

Formularelemente sind in JavaScript Objekte. Das Basisobjekt ist die Seite (document). Bestandteil der Seite ist unter anderem das Formular (*sendform*; deshalb die Benennung). Innerhalb des Formulars wird auf

die Elemente *email* und *name* zugegriffen. Diese Elemente besitzen eine Eigenschaft: *value*, die den Wert zurückgibt.

Der nächste Schritt besteht in der Auswertung der korrekten Werte. Zuerst wird die Länge bestimmt. In JavaScript wird jede Zeichenkette als Objekt betrachtet und ist mit einigen Eigenschaften ausgestattet. Eine davon, *length*, wird zum Test auf die richtige Zeichenanzahl verwendet:

```
if (name.length<3);
```

Zu der Anwendung von *if* ist nichts Neues zu berichten, die Syntax ist mit der von PHP identisch.

```
if ((re_email.test(email))==false);
```

Auf den regulären Ausdruck wird die Methode *test* angewendet. Sie gibt *FALSE* zurück, wenn der Ausdruck unzutreffend ist, sonst *TRUE*. In beiden Fällen wird ein Fehler mit einer kleinen Dialogbox quittiert, die JavaScript mit Hilfe von *alert* erzeugt.

Sind alle Daten richtig eingegeben, wird das *action*-Attribut gefüllt, die Methode eingestellt (hier: *POST*) und das Formular tatsächlich an das PHP-Skript versendet:

```
document.sendform.method = "post";  
document.sendform.action = "<? echo $PHP_SELF ?>";  
document.sendform.submit();
```

Sie können sich die Auswertung von Fehlern mit Hilfe von PHP nun sparen. Der Nutzer kann diesen Test nicht umgehen.

JavaScript für Fortgeschrittene

Die Überprüfung von Formularinhalten ist nicht die einzige sinnvolle Anwendung. Ein anderes Problem entsteht, wenn Sie von einem Formular aus auf mehrere PHP-Skripte zugreifen möchten. Das *<FORM>*-Tag lässt prinzipiell nur ein *action*-Attribut zu, egal ob direkt angegeben oder per JavaScript. Wenn Sie mehrere Ziele benötigen, bieten sich mehrere Formulare an:

```
<FORM action="ziel1.php">  
<INPUT type=submit value="Ziel 1 finden">  
</FORM>  
  
<FORM action="ziel2.php">  
<INPUT type=submit value="Ziel 2 finden">  
</FORM>  
  
<FORM action="ziel3.php">  
<INPUT type=submit value="Ziel 3 finden">  
</FORM>
```

Listing 4.4:
Mehrere Formulare
in einer Seite



Wenn Sie nun aber an alle drei Ziele ein und dasselbe Element senden möchten, tritt ein Konflikt auf. Außerhalb der Formulare sind Elemente unsichtbar oder funktionslos. Wenn Sie ein Eingabefeld in jedes der drei Formulare schreiben, taucht es dreifach auf. Wie das Ergebnis aussehen soll, zeigt Abbildung 4.1, hier gleich mit fünf Schaltflächen.

Abbildung 4.1:
Formular mit
mehreren
Sendeschaltflächen

Ziel ist es, die angegebene E-Mail-Adresse auf Gültigkeit zu prüfen und den Wert an den Server zu übermitteln, der der Schaltfläche entspricht. Gegenüber der Variante mit Optionsschaltflächen (radio) wird ein Mausklick eingespart (der auf die submit-Schaltfläche). Dies kann schon ein Vorteil sein, weil sich die Bedienung vereinfacht.

Die Lösung, die hier gezeigt wird, ist nur eine von vielen Varianten. Die verwendeten Methoden sind aber universell anwendbar:

Listing 4.5:
form_javascript2:
Formulare mit
JavaScript erweitern

```
<html>
<script language="JavaScript">
function senddata(vote) {
    var re_email = /^[_a-zA-Z0-9-]+\.[_a-zA-Z0-9-]+*
        @([a-zA-Z0-9-]+\.)+([a-zA-Z]{2,3})$/;
    var email = document.sendform.email.value;
    if ((re_email.test(email))==true) {
        document.sendform.method = "post";
        document.sendform.vote.value = vote;
        document.sendform.action = "<? echo $PHP_SELF ?>";
        document.sendform.submit();
    } else {
        alert ("E-Mail-Adresse falsch");
    } /* end if */
}
</script>
<body>
<h1>Formulare mit PHP und JavaScript</h1>
<?php
if ($vote) {
    echo "Ihre Bewertung <B>$vote</B> wurde registriert!";
}
?>
<form name="sendform">
Ihre E-Mail-Adresse:<br />
<input type="text" size="40" name="email" />
<input type="hidden" name="vote" value="" />
<P>
```

```

Bitte Ihre Bewertung:<br />
<input type=button name=1 value="Vote 1"
      onclick="senddata(this.name)">
<input type=button name=2 value="Vote 2"
      onclick="senddata(this.name)">
<input type=button name=3 value="Vote 3"
      onclick="senddata(this.name)">
<input type=button name=4 value="Vote 4"
      onclick="senddata(this.name)">
<input type=button name=5 value="Vote 5"
      onclick="senddata(this.name)">
</form>
</body>
</html>

```

Im Beispiel werden wieder reichlich JavaScript-Funktionen benutzt. Die Anwendung des regulären Ausdrucks bedarf sicher keiner weiteren Erläuterung mehr. Interessant ist dagegen, dass auf die `onsubmit`-Methode verzichtet wird. Statt dessen werden normale Schaltflächen verwendet und das Ereignis »Mausklick« auf diese mit `onclick` abgefangen. In ➡ Abschnitt 3.4 *Objektorientierte Programmierung* ab Seite 254 haben Sie bereits die nötigen Grundlagen erlernt. Dort wurde (in PHP) bereits das Schlüsselwort `this` verwendet, um innerhalb einer Methode auf die Methode selbst zu verweisen. Auch JavaScript arbeitet so. Mit `this.name` wird der *Name* der Schaltfläche ermittelt und an die Funktion `senddata` übergeben. Diese Vorgehensweise entkoppelt den Wert (`value`) von der Funktion.

Wie es funktioniert

Im Web sieht man oft die Nutzung der Beschriftung von Schaltflächen für die Skriptsteuerung. Das ist gefährlich! Wenn der Designer die Beschriftung ändert oder an eine andere Sprache anpasst, funktionieren solche Skripte nicht mehr.



Der übergebene Name der Schaltfläche muss nun noch an das PHP-Skript übermittelt werden. Dazu wird ein unsichtbares Feld (`hidden`) genutzt:

```
<INPUT type=hidden name=vote value="">
```

Mit dem folgenden Code wird dem Feld der Name der angeklickten Schaltfläche zugewiesen:

```
document.sendform.vote.value = vote;
```

Das eigentliche PHP-Skript nimmt sich dagegen winzig aus:

```

<?php
if ($vote) {
    echo "Ihre Bewertung <B>$vote</B> wurde registriert!";
}
?>

```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

**PHP
+ JavaScript
+ HTML =
Ganzheitliches
Programmieren!**

Anmerkungen

Der Umgang mit JavaScript ist nicht trivial. Sie sollten sich vor umfangreichen Projekten mit dem nötigen Fachwissen ausstatten. Dieser Abschnitt sollte Sie nur mit dem Nötigsten versorgen, um PHP-Skripte in einer sinnvollen Umgebung entwickeln zu können. Es sollte auch ein wenig der Blick darauf gelenkt werden, wo der Einsatz von JavaScript durchaus sinnvoll ist und dass es hier durchaus auf »ganzheitliches« Denken ankommt. Ebenso sei an dieser Stelle auch nochmals auf die nötigen HTML-Kenntnisse verwiesen. Wenn Sie bei den Beispielen unsicher waren, was die Ausgestaltung der Formularelemente mit Attributen betrifft, lesen Sie unbedingt entsprechende Literatur zu HTML. Sie werden sonst erhebliche Probleme bei den umfangreicheren Beispielen in diesem Buch haben.

4.1.5 Komplexe Formulare

Auch wenn bereits die gezeigten Beispiele nicht trivial waren, in der Praxis werden oft höhere Ansprüche an komfortable Formulare gestellt. Formulare sind die Schnittstelle zum Kunden und wesentlich für den Erfolg einer Website verantwortlich. PHP bietet für alle möglichen Einsatzfälle die nötige Unterstützung.

Formulare mit PHP auswerten

Die bisher gezeigten Beispiele waren vergleichsweise einfach. Es gibt Elemente, die mehr als einen Wert zurückgeben können. So ist es in HTML prinzipiell erlaubt, mehrere Elemente mit dem gleichen Namen zu belegen. Ob gewollt oder nicht, Sie sollten in der Lage sein, solche Namenskonflikte in PHP aufzulösen. Eine typische Anwendung sind mehrere Kontrollkästchen in einer Gruppe:

```
<INPUT type=checkbox name=skill value=php>  
<INPUT type=checkbox name=skill value=asp>  
<INPUT type=checkbox name=skill value=perl>  
<INPUT type=checkbox name=skill value=python>
```

Es gibt auch Fälle, in denen Mehrfachangaben unbedingt erwünscht und unumgänglich sind. So können Sie beispielsweise bei `<SELECT>` mit dem Attribut `multiple` eine Mehrfachauswahl zulassen:

```
<SELECT name=skill multiple size=4>  
  <OPTION value=php>PHP  
  <OPTION value=asp>ASP  
  <OPTION value=perl>Perl  
  <OPTION value=python>Python  
</SELECT>
```

Bei der Auswertung steht Ihnen diese Auswahl nur in einer einzigen Variablen zur Verfügung. Was passiert mit den Werten? Es ist nahe-

liegend, dass PHP dies als Array erkennt. Aber was passiert, wenn Sie die Variable einfach ausgeben?

```
<?php
if ($skill) echo "Sie kennen $skill.";
?>
```

In diesem Fall wird jedoch immer nur der letzte ausgewählte Wert der Liste angezeigt. Offensichtlich wird kein Array gebildet, denn die Auswahl mit `$skill[0]` verweist keineswegs auf den ersten Eintrag der ausgewählten Elemente; die Zeichenkette wird als Array erfasst und der erste Buchstabe wird ausgegeben.

Sie müssen PHP bereits mit dem Namen der Variablen mitteilen, dass `[]` ein Array verwendet werden soll:

```
<SELECT name=skill[] multiple size=4>
```

Sie kennzeichnen den Namen einfach mit eckigen Klammern `[]`, die in HTML keine weitere Bedeutung haben. PHP nimmt nun an, dass mehr als ein Wert folgt und erzeugt ein Array. Dies geschieht natürlich auch dann, wenn tatsächlich nur ein Wert vorliegt. Die Auswertung kann dann komfortabel mit einer Schleife stattfinden:

```
<SELECT name="skill[]" multiple size="4">
  <OPTION value="php">PHP
  <OPTION value="asp">ASP
  <OPTION value="perl">Perl
  <OPTION value="python">Python
</SELECT>
<?php
if (count($skill)) {
  echo "Sie kennen ";
  for ($i=0; $i<count($skill); print($skill[$i++]." ");
}
?>
```

Mit der Funktion `count($skill)` ermitteln Sie die Anzahl der Elemente, die dann einzeln mit `$skill[$i]` adressiert werden. Durch die Angabe des Inkrement-Operators `++` wird zugleich der Zähler erhöht.

Formulare mit PHP erzeugen

Die Angabe der Werte wurde in den vorangegangenen Beispielen stets fest in die Formulare eingebaut. Später werden Sie Datenbankfunktionen kennen lernen, die Werte aus Datenbanken liefern. Es wäre sehr praktisch, wenn die Vorgaben für die Auswahl dynamisch erzeugt werden könnten. Da wir hier noch keine Datenbanken verwenden – es würde die Demonstration nur unnötig komplizieren –, werden die Werte zuerst in Variablen erfasst und dann wieder ausgelesen.

Listing 4.6:
form_selectmultiple:
Auswertung eines
Formulars mit einer
Mehrfachauswahl

*Listing 4.7:
form_selectfill:
Formularelemente in
HTML werden mit
Hilfe von PHP
dynamisch erzeugt*

```
<?php
$skill[] = "PHP";
$skill[] = "ASP";
$skill[] = "Perl";
$skill[] = "Python";
$size = count($skill);
echo "<SELECT name=skill[] multiple size=$size>";
for ($i=0; $i < $size; $i++) {
    echo "<OPTION value=$skill[$i]>$skill[$i]\n";
}
echo '</SELECT>';
?>
```



Wenn Sie assoziative Arrays verwenden, können Sie die Indizes als Werte für die `<OPTION>`-Tags einsetzen und den zugeordneten Wert für die Anzeige verwenden.

Die Liste wird im kompletten Listing 4.7 übrigens auch in der Länge auf die Anzahl der Elemente abgestimmt.



Erweitern Sie die Liste um weitere Einträge und beobachten Sie die Anzeige!

Bei der Anzeige kann es sinnvoll sein, den aktiven Zustand einer Auswahl zu behalten. Wenn Sie die Anzeige aus dem letzten Beispiel beobachten, fällt auf, dass die durch Mausklick erfolgten Markierungen wieder verschwinden. Das `<OPTION>`-Tag kennt das Attribut `selected`, um eine Markierung zu setzen. Die aktivierte Option müsste also durch dynamisches Erzeugen dieses Attributs gekennzeichnet werden. Beim Umsetzen fällt auf, dass mehr auf die Belange von Arrays Rücksicht genommen werden muss als auf die Formelerauswertung. Listing 4.8 zeigt eine praktische Variante:

*Listing 4.8:
form_witharray:
Dynamische
Formularsteuerung
mit assoziativen
Arrays*

```
<html>
<body>
<H1>Formulare mit PHP</H1>
<?php
// Definition
$sskill = array("ph"=>"PHP",
               "ap"=>"Active Server Pages",
               "pl"=>"Perl",
               "pt"=>"Python");

// Ausgabe
if (count($skill)) {
    echo "Sie kennen...<br>";
    while (list($key, $val) = each($skill)) {
        echo "$sskill[$val]<br>";
    }
}
?>
<FORM action="<? echo $PHP_SELF ?>" method=post>
<P>
```

```

Welche Skriptsprachen kennen Sie? <br>
<?php
echo "<select name=\"skill[]\" multiple=\"multiple\"
      size=\"\" . (count($sskill)+1) . \"\">";
echo "<option />Keine";
while (list($key, $val) = each($sskill)) {
    echo "<option value=$key";
    for($i=0;$i<count($skill);$i++) {
        if ($skill[$i]==$key) {
            echo " selected";
        }
    }
    echo " />$val\n";
}
?>
</SELECT>
<INPUT type=submit value="Senden">
</FORM>
</body>
</html>

```

Die Funktionsweise ist simpel. Das Auslesen und Ausgeben des assoziativen Arrays erfolgt mit einer einfachen Schleife, mangels Indizes nicht mit for, sondern mit while:

Wie es funktioniert

```
while (list($key, $val) = each($sskill)) {
```

Innerhalb der Schleife wird gegen das aus den Formularelementen ausgelesene Array *\$skill* geprüft und bei Übereinstimmung wird einfach das Attribut *selected* ausgegeben:

```

for($i=0;$i<count($skill);$i++) {
    if ($skill[$i]==$key) {
        echo " selected";
    }
}

```

Globale Servervariablen

Wenn Sie nicht genau wissen, wie die Felder heißen, bietet sich ein Zugriff auf alle übermittelten Formularvariablen an. Die Bestätigung aus dem Beispiel in Listing 4.2 könnte dann auch so aussehen:

**\$HTTP
_POST_VARS**

```

<?php
if ($sent) {
    echo "Sie haben folgende Daten gesendet:<P>";
    while (list($key, $val) = each($HTTP_POST_VARS)) {
        echo "<B>$key</B>: $val<BR>\n";
    }
}
?>

```

Formulardefinition aus Listing 4.2 (siehe Seite 305)

Listing 4.9:
form_postvars:
Auslesen aller Form-
Variablen aus
\$HTTP_POST_VARS

`$HTTP_POST_VARS` ist ein assoziatives Array, das als Indizes die Namen aller Formularelemente und als Werte deren Inhalte enthält.

Vorsicht Falle!

Kritische Benennung der Variablen

In diesem Zusammenhang kann noch ein Problem auftreten, das im kompletten Beispiel aus Listing 4.9 unterschlagen wurde. Bei der Auswertung der Formularelemente wird der Inhalt in ein Array mit dem Namen `$skill` transportiert. Wenn Sie mit `$skill[]` nun weiter unten Elemente hinzufügen, wird die Liste immer länger, und zwar um die zuvor ausgewählten Elemente. In Listing 4.7 wird das Array deshalb mit `$skill=""` zerstört. Diese Methode sollte nur zur Demonstration dienen, in der Praxis sollten Sie Ihre Variablen sorgfältiger benennen und für die Verwendung in PHP andere Name verwenden als für die entsprechenden Elemente in HTML.

Umgang mit <textarea>

Auch im Zusammenhang mit `<textarea>` können Probleme auftreten, wenn der eingegebene Text anderweitig zur Anzeige gebracht werden soll. Zeilenumbrüche innerhalb des erfassten Textes entsprechen dem ASCII-Standard und werden in PHP mit dem Code `\n` dargestellt. Um die Darstellung nicht zu verfälschen, wäre in HTML aber eine Ausgabe mit `
` erwünscht. Sie können dafür eine einfache Textersetzung verwenden oder die speziell dafür entworfene Funktion `n12br` nutzen:

*Listing 4.10:
Ausschnitt aus
form_textarea:
Zeilenumbrüche in
<textarea> werden
in HTML-Umbrüche
verwandelt.*

```
<?php
if ($sent & $name) {
    $message = n12br($message);
    echo <<<FORMANSWER
    # Fortsetzung wie in Listing 4.2 (siehe Seite 305)
```

4.2 Daten per URL weiterreichen

Neben POST wird auch die Methode GET zum Übertragen von Daten verwendet – durch Anhängen der Werte an den URL. Dieser Abschnitt beschreibt, wie Sie Skripte per Hyperlink »verbinden«.

4.2.1 Wie werden Daten weitergereicht?

Die bisher bei den Formularen gezeigten Methoden zur Übergabe von Daten wurden zwar immer wieder auf dasselbe Skript zurückgeführt, dies diente jedoch nur der Vereinfachung. Ebenso könnte man von einem Skript zum Nächsten springen und dabei Daten mitnehmen. Eine andere Möglichkeit der direkten Übergabe von Variablen zwischen Skripten besteht nicht, wenn man von »externen« Varianten wie gespeicherten Textdateien oder Datenbanken einmal absieht.

Im ➡ Abschnitt 4.1.1 *GET und POST* (Seite 301) wurde die Methode GET bereits vorgestellt, die Daten durch anhängen an den URI über-

trägt. Es ist naheliegend, auf diese Methode nicht nur mittels Formularen, sondern direkt zuzugreifen. Dieser direkte Zugriff erfolgt mittels entsprechend gestalteter Links:

```
<a href="skript.php?var=wert&var2=wert">Sprung</a>
```

Drei spezielle Zeichen finden dabei Verwendung. Das Trennzeichen zwischen URL und Daten ist das Fragezeichen (?). Die Trennung der einzelnen Variablen-/Wertepaare ist das Ampersand (&). Die Trennung innerhalb der Paare erfolgt mittels Gleichheitszeichen (=). Beachten Sie, dass Browser nur ca. 2 000 Zeichen akzeptieren, einschließlich URL. Größere Datenmengen können tatsächlich nur mit Formularen übertragen werden.

Eine Extrahierung der Daten ist ebenso wie bei den Formularen nicht notwendig. Alle Daten stehen in einzelnen Variablen zur Verfügung. Eine typische Anwendung zeigen die folgenden beiden Skripte:

```
<h3>Was möchten Sie bestellen?</h3>
<a href="get_linkanswer.php?artikel=1">Der Unbesiegbare</a><br>
<a href="get_linkanswer.php?artikel=2">Der Schnupfen</a><br>
<a href="get_linkanswer.php?artikel=3">Sternstagebücher</a>
<br>
<a href="get_linkanswer.php?artikel=4">Eden</a><br>
```

*Listing 4.11:
get_link: Übertragung von Daten per Hyperlink*

Im Beispiel werden Artikeldaten in einer Variablen `$artikel` verpackt und mit dem Link an die aufgerufene Seite übertragen (`get_linkanswer` in Listing 4.11).

```
<h3>Was haben Sie bestellt?</h3>
Sie haben das Buch <i>
<?php
switch($artikel)
{
    case 1: echo "Der Unbesiegbare "; break;
    case 2: echo "Der Schnupfen "; break;
    case 3: echo "Sternstagebücher "; break;
    case 4: echo "Eden ";
    }
?>
</i>von Stanislaw Lem bestellt. Vielen Dank!
```

*Listing 4.12:
get_linkanswer: Die Antwort zu Listing 4.11: Die Variable \$artikel wird automatisch aus der URL extrahiert.*

Das war sicher sehr einfach. Praktisch werden solche Weitergaben sehr häufig eingesetzt. Oft dienen Bilder als Schaltflächen und es ist dann einfacher, mit `<A>`-Tags zu arbeiten anstatt mit Formularen.

Kodierung von Daten

Solange Sie nur Artikelnummern übertragen, wird diese Form problemlos funktionieren. Betrachten Sie das folgende Beispiel aufmerksam:

```
<a href="get_coded.php?email=Ihr Name?">Ihr Name?</a>
```


Wenn der Eintrag »Ihr Name?« beispielsweise aus einem Formular übernommen wurde, kann das Skript nicht richtig funktionieren. Leerzeichen sind an dieser Stelle nicht zulässig. Die Daten müssen also auf ein Format gebracht werden, in dem die Sonderzeichen (? & =) tatsächlich als Sonderzeichen erkannt werden.

Die nötigen Funktionen zur Kodierung und Dekodierung haben Sie bereits im ➡ Abschnitt *Codierung von URL-Daten* ab Seite 200 kennen gelernt.

**urlencode()
urldecode()**

Mit `urlencode` ersetzen Sie diese Zeichen durch die Zeichenfolge `%HH`, wobei `HH` den Hexcode des betreffenden Zeichens im ASCII-Zeichensatz darstellt. `urldecode` wandelt eine so kodierte URL wieder zurück. Wenn Sie eine URL bilden, schreiben Sie deshalb besser folgenden Code:

```
$coded = urlencode($email);
```

Dann setzen Sie den Wert in den Link ein:

```
echo "<a href=\"get_coded.php?email=$coded\">";  
echo "Ihr Name?</a>";
```

Wenn Sie für den Text der Variablen folgende Zeichen einsetzen,

```
$email = "Jörg's E-Mail?";
```

finden Sie im Quelltext diese Form:

```
J%F6rg%27s+E-Mail%3F
```

Die ersetzten Zeichen wurden für den Satz hervorgehoben. Das Leerzeichen wird durch ein Pluszeichen ersetzt, das Pluszeichen durch den entsprechenden ASCII-Code. Hier das Listing:

*Listing 4.13:
get_coded: Daten via
URL übermitteln*

```
<?php  
echo "$email wurde &uuml;bertragen";  
$email = "Jörg's E-Mail?";  
$coded = urlencode($email);  
echo "<p><A HREF=\"$PHP_SELF?email=$coded\"><b>$email</b>  
echo "&uuml;bertragen ?</A>";  
?>
```

4.2.2 Probleme

Bei der Kodierung und Dekodierung können verschiedene Probleme auftreten, vor allem, wenn die Daten mehrfach weiterverarbeitet werden.

Störende Escape-Zeichen

php.ini

Wenn Sie die Daten im Beispiel in Listing 4.13 um Anführungszeichen ergänzen zeigt sich, dass es versteckte Probleme geben kann. PHP hat

im Abschnitt [Data Handling] der Konfigurationsdatei PHP.INI den folgenden Eintrag:

```
magic_quotes_gpc = On
```

In diesem Fall werden einfache und doppelte Anführungszeichen mit dem Escape-Zeichen (\, Backslash) versehen. Auch der Backslash selbst wird so gekennzeichnet, als doppelter Backslash (\\) also.¹⁴ Die Zeichenfolge »Jörg's "E-Mail?"« sieht dann so aus:

```
Jörg\'s \'E-Mail?\'
```

Wenn Sie mit Datenbanken arbeiten, ist dieser Effekt erwünscht, denn oft werden zeichenbasierte Daten mit Anführungszeichen umgeben. Für die Ausgabe in HTML stört das. Sie müssen den Escape-Effekt ausschalten. Wenn Sie generell diese Funktion nicht nutzen, lohnt die Deaktivierung in der Datei PHP.INI:

```
magic_quotes_gpc = Off
```

Wollen Sie nur gelegentlich die Angabe der Escape-Zeichen unterdrücken, können Sie die Funktion `stripslashes()` einsetzen, wie im letzten Beispiel bereits erfolgt:

```
echo stripslashes($email). " wurde &uuml;bertragen";
```

Ein anderes Problem tritt auf, wenn Sie Formularfelder und Variablen im URL mit denselben Namen verwenden. In solchen Fällen wird vorrangig die URL dekodiert. Sie können dennoch auf den Inhalt »beider« Variablen zugreifen.

Mit \$QUERY_STRING arbeiten

Zum einen steht die globale Variable `$QUERY_STRING` zur Verfügung, die den gesamten Teil nach dem `?`-Zeichen enthält. Hier müssen Sie sich zwar selbst um die Analyse der einzelnen Elemente kümmern, umgehen aber die automatische Auswertung des PHP-Parsers. Da Ihnen der Zugriff auf die Formularvariable verwehrt ist, bleibt auch hier die Nutzung der Servervariablen. Das folgende Beispiel zeigt die Anwendung und Ausgabe:

```
<?php
echo "&Uuml;bertragen wurde sent = $sent<P>";
echo "$QUERY_STRING<P>";
if (is_array($_POST)) {
    while (list($key, $val) = each($_POST)) {
        echo "<B>$key</B>: $val<BR>\n";
    }
}
```

*Listing 4.14:
get_querystring:
URL-Daten und
Formularelement mit
identischem Namen
unterscheiden*

¹⁴ Verändert wird auch das Null-Byte NUL, was aber bei Zeichenketten nur sehr selten in Betracht kommt.

```

}
?>
<form action="<?php echo $PHP_SELF ?>?sent=yes" method="post">
<input type=hidden name=sent value=no>
<input type=submit>
</form>

```

Wie es funktioniert Sie sehen an diesem Beispiel auch, wie Sie die URL zur Datenübertragung auch mit einem Formular verwenden können. Interessant ist die Reaktion bei der Anwendung von GET im Formular:

```

<form action="<? echo $PHP_SELF ?>?sent=yes" method=get>
<input type=hidden name=sent value=no>
<input type=submit>

```

In diesem Fall nehmen die Formularelemente den Platz der Daten im URL ein, die bereits im action-Attribut gezeichneten Parameter werden ignoriert. Dieses Verhalten hat nichts mit PHP zu tun, sondern basiert auf HTTP und den Vorschriften zur Verarbeitung von Daten mit Hilfe der Methoden GET und POST.

Beim Weiterreichen von Daten ist die Anwendung von `$QUERY_STRING` sogar besonders bequem. Wenn Sie auf Cookies (➡ Abschnitt 4.3.1 *Einführung in Cookies* ab Seite 324) verzichten müssen und eine Anwendung erstellen, die aus mehreren Skripten besteht, ist eine »Verfolgung« des Nutzers unerlässlich. Es werden dann meist so genannte Session-IDs verwendet – per Zufallsgenerator erzeugte Nummern oder Zeichenfolgen, die den Nutzer immer wieder eindeutig zuordnen. Auf der ersten Seite wird die Nummer einer Variablen übergeben:

Listing 4.15:
get_queryform: Wie-
tergabe von Daten
über die Variable
`$QUERY_STRING`

```

<?php
if (!isset($session)) {
    $session = 1;
    $QUERY_STRING = "session=".$session;
    echo "\$QUERYSTRING nicht ge&auml;ndert<p />";
} else {
    echo "\$QUERYSTRING ge&auml;ndert<p />";
}
echo <<<FOOTER
    <a href="$PHP_SELF?$QUERY_STRING">Link mit $QUERYSTRING</a>
    <p />
    <a href="$PHP_SELF">Link ohne $QUERYSTRING</a>
FOOTER;
?>

```

Gleichlautende Variablen

Bei der automatischen Generierung von Daten kann es vorkommen, dass mehrere gleichlautende Variablen unterschiedliche Werte enthalten:

```
<a href="zielskript.php?x=1&x=2&x=3">Link</a>
```

In diesem Fall extrahiert PHP nur die letzte Variable und den dazugehörigen Wert. Die Ausgabe von `echo $x;` zeigt in diesem Beispiel »3« an. Wenn Sie sich jedoch alle Daten ansehen, so ist zu erkennen, dass tatsächlich alle Daten übertragen werden:

```
echo $QUERY_STRING;
```

Sie müssten in diesem speziellen Fall die Daten selbst mit Hilfe der Funktion `explode` dekodieren. `explode` zerlegt eine Zeichenkette anhand eines Trennzeichens in ein eindimensionales Array (Listing 4.15). Anschließend erfolgt die nochmalige Trennung anhand des Gleichheitszeichens und (hier zu Demonstrationszwecken) die Ausgabe in einer `while`-Schleife:

```
<?php
$pairs = explode("&", $QUERY_STRING);
for ($i = 0; $i < count($pairs); $i++) {
    $query[$i] = explode("=", $pairs[$i]);
}
while (list($key, $val) = each($query)) {
    echo "Parameter: <b>$val[0] </b>: $val[1]<br>";
}
?>
```

explode()

*Listing 4.16:
get_pairs: Mehr-
faches Auftreten
gleichnamiger Vari-
ablen muss selbst
analysiert werden*

Die Vorgehensweise ist hier nicht besonders elegant. Vermeiden Sie gleichnamige Variablen, wo immer Sie können.

Lange Parameterschlangen

Wenn Sie viele Parameter mit GET übergeben müssen, kann der entsprechende Befehl sehr unübersichtlich werden. Korrekturen oder Änderungen gestalten sich entsprechend kompliziert. Ein interessanter Trick ist die Bildung der URL mit der `sprintf`-Anweisung. Dazu fertigen Sie zuerst eine Vorlage, die alle Variablen der URL enthält. Dann werden die Werte mit `sprintf` zugewiesen und später ausgegeben. Alternativ kann man natürlich auch gleich `printf` verwenden.

```
<?php
$text_to_show = 'Klick mich!';
$wert1 = 1; $wert2 = 2; $wert3 = 3;
$get = '<a href="%s?wert1=%s&wert2=%s&wert3=%s">%s</a>';
$openlink = sprintf($get, $PHP_SELF,
                    urlencode($wert1), urlencode($wert2),
                    urlencode($wert3), $text_to_show);

echo $openlink;
echo '<br>';
echo htmlspecialchars($openlink);
?>
```

*Listing 4.17:
genlinks: Erzeugen
von Links*

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

So behalten Sie leicht die Übersicht. Mit Hilfe der Funktion `urlencode` stellen Sie sicher, dass jeder beliebige Inhalt einer Variablen übertragen wird. `$nextfile` enthält den Namen des aufzurufenden Skripts.

Bei sehr umfangreichen Variablensammlungen können Sie auch Arrays einsetzen. Beachten Sie grundsätzlich, dass die Browser als maximale Länge der URL etwa 2 000 Zeichen zulassen. Dabei zählen außer den Variablennamen auch der jeweilige Inhalt und die Trennzeichen mit.

4.3 Cookies

Cookies sind für jeden Webprogrammierer eine der Basisfunktionen. Um sie erfolgreich nutzen zu können, ist ein wenig Grundlagenwissen sinnvoll. Neben der technischen Nutzung wird in diesem Abschnitt auch auf die Geschichte eingegangen.

4.3.1 Einführung in Cookies

Cookies sind ebenso hilfreich wie verrufen. Warum wird über diese Technik so kontrovers diskutiert?

Kennen Sie Cookies?



Cookies (deutsch: Kekse) haben einen völlig harmlosen, irreführenden Namen. Sie sind bekannt als Angriffspunkt der Hacker und Spionagelinstrument gegen den unwissenden Surfer. Beides ist falsch.

Die Ursache für den ganzen, ich will es vorwegnehmen, unbegründeten Ärger ist ein Artikel von Jon Udell in der Märzangabe 1997 der Zeitschrift BYTE, einer der größten amerikanischen Computerefachzeitschriften. Dort wurde berichtet, dass Cookies Informationen auf Geheiß des Servers auf der lokalen Festplatte des Nutzers speichern und natürlich auch lesen können. Daraus wurde geschlußfolgert, dass der Server private Daten vom Computer des Surfers lesen kann. Des Weiteren wurde behauptet, dass andere Server wiederum die Daten lesen können, die schon im Cookie gespeichert sind, wodurch das Auslesen privater Kennwörter möglich sein soll. All das hat zur gnadenlosen Vorverurteilung der Cookies beigetragen und dazu geführt, dass viele Nutzer die Funktion abschalten oder sogar die Cookiedatei regelmäßig löschen. In der Maiausgabe 1997 der BYTE wurde der gesamte Artikel revidiert und richtiggestellt – zu spät, wie sich herausstellte. Die sensationslüsterne Presse hatte wieder etwas Negatives am Web entdeckt und wollte sich das Spielobjekt nicht wegnehmen lassen. Der Pro-Cookie-Artikel wurde totgeschwiegen.

Netscapes Cookie-Definition

Netscape gilt als Erfinder der Cookies. Die Firma wurde durch ihren Browser Navigator bekannt. Mit den Cookies sollten die Unzulänglichkeiten des HTTP umgangen werden.



Cookies sind ein grundsätzlicher Mechanismus, den serverseitige Verbindungen (wie CGI-Skripte) nutzen können, um clientseitig Informationen zu speichern und wieder auszulesen.

Was sind Cookies?

Wenn ein Server Informationen in Form eines HTTP-Objekts an den Client zurücksendet, kann er zusätzlich ein Informationsstück liefern, das der Client speichert. Verbunden mit diesem Stück ist eine Beschreibung einer Reihe von URLs, für die dieses Informationsstück gültig ist. Jede zukünftige Anforderung via HTTP von diesem Client an eine der gespeicherten URLs enthält das gespeicherte Stück Information. Diese Information wird Cookie genannt.

Dieser einfache Satz aus der Netspace-Definition sagt auch klar, warum die Angst vor Cookies unbegründet ist. Nicht der Server liest Informationen aus der Cookie-Datei aus, sondern der Client (Browser) sendet diese (und nur diese) Informationen an die URL zurück, von der sie auch geschrieben wurden. Da der Server dabei passiv agiert, kann er diesen Prozess auch nicht manipulieren – theoretisch. Lesen Sie dazu auch den ➡ Abschnitt *Wie Sie beim Surfen mit Cookies verfolgt werden* ab Seite 328.

Ist die Angst begründet?

Die Anwendung ist vielfältig. Shopprogramme könnten den Warenkorb, gebührenpflichtige Seiten die Anzahl der Besuche speichern und personalisierte Seiten erkennen den Nutzer wieder und registrieren die Zeit, wann er zuletzt da war.

Ein Cookie wird an den Client gesendet, indem im Kopf der HTTP-Antwort ein Set-Cookie-Header eingesetzt wird. Typischerweise wird dieser durch ein CGI-Skript erzeugt.

Spezifikation

Die folgende Darstellung zeigt, wie ein solches Cookie im HTTP aussieht:

Syntax des Set-Cookie-Header

```
Set-Cookie: NAME=VALUE; expires=DATE; path=PATH;
domain=DOMAIN_NAME; secure
```

```
NAME=VALUE
```

Beschreibung des Cookies

Dieses ist der Name und Inhalt des Cookies und der einzige Parameter, der nicht optional ist. Erlaubt sind alle Zeichen außer Kommata, Semikola und Leerzeichen. Es ist empfehlenswert, Funktionen wie `urlencode` zu verwenden, wenn mit dem Inhalt umfangreichere Daten gespeichert werden sollen.

```
expires=DATE
```

Das Verfallsdatum definiert, bis zu welchem Zeitpunkt das Cookie leben darf. Wird das Verfallsdatum überschritten, wird das Cookie gelöscht und nicht mehr an den Server gesendet.

Das Datum muss folgendes Format haben:

```
Wdy, DD-Mon-YYYY HH:MM:SS GMT
```

Die Darstellung basiert auf RFC 822, RFC 850, RFC 1036 und RFC 1123. Die einzige legale Zeitzone ist GMT. Beachten Sie die Minuszeichen als Trennzeichen im Datum – dies entspricht nicht den Möglichkeiten, wie sie in den genannten RFCs dargestellt werden.

Der Parameter `expires` ist optional. Wird er nicht angegeben, verfällt das Cookie am Ende der Sitzung.

```
domain=DOMAIN_NAME
```

Wenn der Browser in der Liste der gespeicherten Cookies sucht, vergleicht er das Domainattribut `domain` mit dem Internet-Domainnamen des Servers, von dem die URL aufgerufen wurde. Dabei werden auch Teile einer Domain berücksichtigt. Der Domainname »acme.com« wird also auch für vollständige Domains der Art »shipping.crate.acme.com« gültig sein. Nur Server, deren Domainname übereinstimmt, können Cookies setzen und lesen. Außerdem muss der Domainname wenigstens drei Punkte enthalten, ausgenommen davon sind die folgenden generischen Topleveldomains, bei denen zwei Punkte genügen:

- COM
- EDU
- NET
- ORG
- GOV
- MIL
- INT

Wird das Attribut nicht angegeben, wird der Name des Servers (Host-name) verwendet.

```
path=PATH
```

Das Pfad-Attribut wird verwendet, um eine Untereinheit der URL innerhalb einer Domain angeben zu können. Wenn der Domainname überprüft und eine Übereinstimmung gefunden wurde, wird nun der Pfad verglichen. Stimmt auch der Pfad überein, wird das Cookie mit der nächsten Anforderung an den Server übertragen.

Der Pfad »/foo«¹⁵ wird als gültig angesehen für »/foobar« und »/foo/bar.html«. Der Pfad »/« ist ein genereller Platzhalter. Wird kein Pfad angegeben, wird der Pfad der aufgerufenen Seite angenommen.

secure

Wenn ein Cookie mit dem Attribut `secure` gekennzeichnet ist, wird es nur übertragen, wenn die Verbindung über einen sicheren Kanal zustande kommt (HTTPS, das heißt HTTP über SSL). Ohne dieses Attribut werden Cookies unverschlüsselt (Klartext) übertragen.

Wenn ein URL von einem Server via HTTP angefordert wird, prüft der Browser die URL gegen alle Cookies und fügt der Anforderung an den Server bei Übereinstimmung Variablen/-Wertepaare hinzu, die in die Anforderung eingeschlossen werden. An dieser Stelle wird klar: Ein Server fordert keine Cookies und liest keine Cookies, er bekommt sie vom Browser geliefert!

**Syntax der Cookie
HTTP Request
Header**

Diese Paare haben das folgende Format:

```
Cookie: NAME1=OPAQUE_STRING1; NAME2=OPAQUE_STRING2 ...
```

Mehrere Cookies werden in einer Anforderung übertragen. Schreiben zwei Server mit demselben Namen und Pfad Cookies, überschreibt der letzte Prozess den vorhergehenden ohne Rückfrage oder Benachrichtigung. Wird ein anderer Name verwendet, wird dieser als weiteres Cookie der Liste hinzugefügt.

**Zusätzliche
Hinweise**

Wird der Pfad auf eine weitere Verzeichnistiefe erweitert, so wird ein neues Cookie erzeugt. Treffen nun bei der Abfrage mehrere Cookies zu, werden alle zutreffenden Übereinstimmungen gesendet.

Das Verfallsdatum teilt dem Client lediglich mit, ab wann es sicher ist, dass das Cookie gelöscht werden kann. Es gibt aber keinen Zwang, dass zu diesem Zeitpunkt tatsächlich die Löschung erfolgt. Ebenso können Cookies vor Ablauf des Verfallsdatums gelöscht werden, wenn der intern dafür bereitgestellte Speicherplatz erschöpft ist.

Beim Senden der Cookies werden die weiter übereinstimmenden Pfadangaben *vor* den weniger übereinstimmenden übertragen.

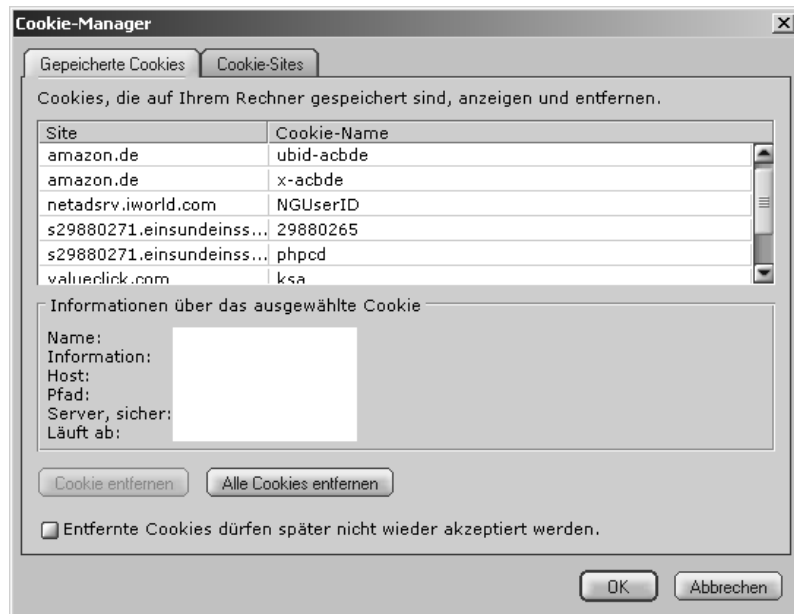
Die maximalen Werte für Cookies, die ein Browser speichert, betragen: **Cookie-Grenzen**

- 300 Cookies insgesamt
- 4 KByte pro Cookie (Name und Inhalt zusammen)
- 20 Cookies pro Server und Domain

¹⁵ Die Bezeichner »foo« oder »bar« sind Hackerslang und bezeichnen irgendwas, was sonst noch keinen Namen hat, oder stehen als Platzhalter für Realnamen.

Wenn diese Grenzen überschritten werden, darf der Client das am längsten ungenutzte Cookie löschen. Wenn das Cookie größer als 4 KByte ist, wird der Inhalt gekürzt, nicht aber der Name.

Abbildung 4.2:
Der Cookie-Manager
im Netscape
Navigator 6



Wenn der Server ein Cookie explizit löschen will, sendet er ein schon vorhandenes Cookie (vorhandener Name) mit einem Verfallsdatum, das in der Vergangenheit liegt. Pfad und Name müssen exakt übereinstimmen.

HTTP-Proxys und andere Zwischenspeicher sollen Set-Cookie-Header nicht speichern. Wenn ein Set-Cookie-Header auftritt, wird dieser auch dann weitergereicht, wenn der Server mit 304 (Not Modified) oder 200 (OK) geantwortet hat. Normalerweise würden Seiten in diesem Fall nicht erneut gesendet, sondern aus dem Cache geladen werden.

Wie Sie beim Surfen mit Cookies verfolgt werden



Wenn Sie sich die Cookie-Datei aufmerksam ansehen, werden Sie mit hoher Wahrscheinlichkeit ein Cookie entdecken, das *doubleclick* oder der deutschen Entsprechung *doppelklick* gehört. Ebenso hoch ist die Wahrscheinlichkeit, dass Sie noch nie auf der Website unter *www.doubleclick.net* waren. Wie kommt nun dieses »fremde« Cookie in den Browser? Ein Blick auf die Seite der Firma Doubleclick offenbart die Motivation. DoubleClick liefert Marktinformatonen über die Nutzer des Internet in Echtzeit. Dazu werden von jedem Nutzer Profile angelegt. Die Erkennung der richtigen Nutzer erfolgt – logischerweise – per Cookie. Aber wie funktioniert das? Wenn niemand die Double-

Click-Site besucht, können die Cookies nie gesetzt werden – oder doch?

DoubleClick-Kunden sind Händler und Anbieter im Internet, die mehr über ihre Kunden wissen möchten. Wenn ein argloser Surfer auf die Homepage eines DoubleClick-Kunden tritt, wird er kurz an DoubleClick umgeleitet, wo das Cookie gesetzt wird, und weiter geht die Reise, diesmal mit DoubleClick-Cookie im Gepäck. Jede andere Seite mit Verbindung zu DoubleClick verfährt ebenso, nur liefert der Browser brav die alten DoubleClick-Cookies aus – eine perfekte Verfolgung über alle Sites, die im DoubleClick-Netzwerk enthalten sind. Nebenbei liefern die DoubleClick-Kunden auch Daten über die erkannten Nutzer und füllen damit die Profile, im Gegenzug erhalten Sie ständig verbesserte Profile. Das Profil selbst wird bei DoubleClick gespeichert – das Cookie enthält nur die ID-Nummer (siehe Abbildung 4.2). Genutzt werden die Profilinformationen für die Steuerung der Werbung. So werden zwei Surfer, die zur gleichen Zeit die gleiche Seite besuchen, verschiedene Banner sehen – je nach Profil.

Das eigentliche Problem ist nicht die Tatsache, dass auf diese Weise Informationen gesammelt werden, sondern dass dies heimlich und ohne Kenntnis der Nutzer erfolgt. Erst beim Abschalten der Cookies bemerken viele Surfer, mit welcher Intensität beim Surfen davon Gebrauch gemacht wird. Diese Nutzung entspricht im Übrigen nicht mehr der ursprünglichen Intention. Die Zwischenschaltung des DoubleClick-Servers lässt die Cookies transparent werden – genau das sollte per Definition aber nie passieren.

Natürlich lassen sich Cookies abschalten. Aber dies funktioniert bei allen Browsern nur global – alles oder nichts. Schade eigentlich, denn manchmal ist die persönliche Begrüßung oder eine gespeicherte persönliche Einstellung angenehm und erleichtert den Umgang mit manchen Sites.

4.3.2 Cookies in PHP

Cookies werden in PHP durch eine einzige Funktion unterstützt. Dies ist in den meisten Fällen ausreichend. Wie schon bei Formularen und GET-Daten fragt PHP alle Daten im Header des HTTP-Request automatisch ab und erstellt entsprechend gefüllte Variablen. Die `setcookie()`-Funktion hat folgenden Aufbau:

```
setcookie("Name", "Inhalt", Verfallsdatum, Pfad, Domain, secure);
```

Außer dem Namen "Name" sind alle Argumente optional. Wenn Sie Parameter weglassen, müssen Sie die Angaben Pfad und Domain durch leere Zeichenketten "" ersetzen, Verfallsdatum und secure durch 0.

Der "Inhalt" wird automatisch mit `urlencode` behandelt. Sie müssen diese Funktion nicht explizit anwenden. Entsprechend erfolgt auch automatisch eine Rückumwandlung der Cookies.

Anwendungsbeispiele

Eine einfache Anwendung wäre die Speicherung der Farbe einer Seite. Das folgende Skript fragt beim ersten Start die gewünschte Seitengestaltung ab, speichert diese dann in einem Cookie und präsentiert die Farbe beim nächsten Aufruf automatisch. Das Cookie verfällt nach drei Minuten (180 Sekunden).

Listing 4.18:
cookie_setcontrol:
Cookies steuern die
Seitenfarbe

```
<?php setcookie("color",$setcolor,time()+180); ?>
<h3>Personalisierte Seite</h3>
<?
if (!$color) {
    echo "Cookie nicht gesetzt";
} else {
    echo "Cookie sagt: Farbe $color";
}
?>
<P>
<form method="post" action="<? echo $PHP_SELF ?>">
<select name=setcolor size=1>
    <option value=>Farbauswahl
    <option value=red>Rot
    <option value=green>Grün
    <option value=blue>Blau
</select>
<P>
<input type=submit>
</form>
```

Wie es funktioniert Zuerst einmal muss der in ➡ Abschnitt 4.3.1 *Einführung in Cookies* ab Seite 324 gezeigten Beschreibung entsprochen werden. Cookies werden im Kopf der HTTP-Antwort gesendet. Der Seiteninhalt kann also erst danach geschrieben werden. Deshalb steht der Sendebefehl *vor* dem ersten HTML-Tag (<HTML>):

```
<? setcookie("color",$setcolor,time()+180); ?>
```

Die Abfrage, ob das Cookie bereits gesetzt wurde, erfolgt folgendermaßen:

```
if (!$color)
```

Wenn das Cookie nicht existiert, ist der Wert `FALSE`, und mit der Negation (!) wird der erste Teil der Bedingung ausgeführt. Die Übertragung der Daten in das Cookie erfolgt mit dem Ausführen der Seite. Der erste Aufruf geht dabei noch ins Leere: Zuerst muss das Formular abgesendet werden; beim zweiten Mal wird dann das Cookie ge-

schrieben und beim dritten Mal kann es – bis zum Ende des Verfallsdatums immer wieder – gelesen werden.

Die simple Vorgehensweise führt zu einem merkwürdigen Verhalten. Sie müssen, um tatsächlich einen Farbwechsel hervorzurufen, zweimal auf Senden klicken. Alternativ können Sie beim zweiten Mal auch den Refresh- oder Aktualisieren-Schalter im Browser anklicken.

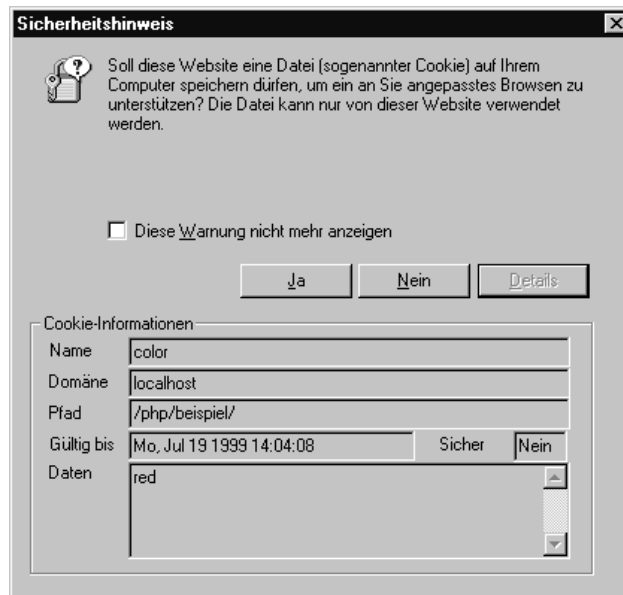


Abbildung 4.3:
Der Internet
Explorer hat das
Cookie erkannt und
zeigt dem Nutzer die
Daten an (Cookie aus
Listing 4.18)

Wenn Sie größere Datenmengen in Cookies speichern, können Sie Arrays verwenden und mit den Befehlen `implode` und `explode` von und in Zeichenketten umwandeln.

```
<?php
function setArray($cookieArray, $name, $expires) {
    $cookstring = implode("@@@", $cookieArray);
    setcookie($name, $cookstring, time()+3600);
}
function getArray($name) {
    return explode("@@@", $name);
}
$myarray[] = "Farbe";
$myarray[] = "Name";
$myarray[] = "Ort";
$myarray[] = "Interesse";
$myarray[] = "eMail";
setArray($myarray, "profile", time()+86400);
?>
```

Listing 4.19:
cookie_setarray
Arrays in Cookies
speichern

Listing 4.20:
cookie_getarray: In
Cookies gespeichertes
Array wieder aus-
lesen (Antwortsript
zu Listing 4.19)

```
<?php
function getCArray($name) {
    return explode("@@@", $name);
}
?>
<html>
<body>
<h1>Cookies verwenden</h1>
<H3>Arrays aus Cookies auslesen</H3>
<?
$myarray = getCArray($profile);
for ($i=0; $i<=count($myarray); $i++) {
    echo "$myarray[$i]<BR>";
}
?>
</body>
</html>
```

Namenskonflikte

php.ini

Da das Auslesen der Cookies durch Übergabe des Inhalts in eine gleichlautende Variable erfolgt, kann es Namenskonflikte mit GET- oder POST-Daten geben. Welche Datenquelle gewinnt, kann in der Datei PHP.INI eingestellt werden. Suchen Sie den folgenden Eintrag im Abschnitt [Data Handling]:

```
gpc_order = gpc;
```

Wenn der Eintrag nicht existiert, fügen Sie ihn hinzu. Die Reihenfolge wird durch das Argument bestimmt, g steht für GET, p für POST und c für Cookie. Mit gpc ist also die Reihenfolge GET→POST→Cookie gemeint.



Benennen Sie Cookies konsequent nach einem bestimmten Schema, beispielsweise "cook_xxxxx". Dann können Namenskonflikte sicher vermieden werden.

\$HTTP_COOKIE_VARS

Wenn diese Methoden keine ausreichende Flexibilität bieten, können Sie die Cookies auch direkt aus dem assoziativen Array \$HTTP_COOKIE_VARS auslesen. Das folgende Beispiel zeigt alle Cookies an:

Listing 4.21:
readallcookies:
Auslesen aller
Cookies

```
<?php
foreach($HTTP_COOKIE_VARS as $key => $val) {
    echo "<B>$key</B> = $val<BR>\n";
}
?>
```

Alternativ kann auch while in Kombination mit list und each verwendet werden, um auf die Elemente des Arrays zuzugreifen.

Wenn Sie diese Schleife in eine Funktion einbauen, vergessen Sie nicht, die globale Variable `$HTTP_COOKIE_VARS` lokal verfügbar zu machen:

```
global $HTTP_COOKIE_VARS;
```



4.3.3 Cookies und JavaScript

Im Zusammenhang mit Cookies tauchen immer wieder Fragen zur Nutzung von JavaScript auf. Vor allem Kombinationen aus PHP und JavaScript erscheinen ausgesprochen leistungstark. Das ist, prinzipiell, auch richtig. Nur haben die von und mit JavaScript gesetzten Cookies nichts mit den vom Server (ob mit PHP oder Perl spielt keine Rolle) gesetzten Cookies zu tun.

Sie können grundsätzlich keine Cookies auslesen, die vom jeweils anderen System gesetzt wurden. JavaScript-Cookies speichern bestimmte Werte in der Cookie-Datei des Browsers. Dies ist nur eine von Netscape aus Bequemlichkeit verwendete Speicherform. Prinzipiell ist es bloß eine Textdatei, die Werte speichert und auf Anforderung wieder ausgibt.

Cookies können vom Nutzer ausgeschaltet werden. Dies betrifft auch die von JavaScript gesetzten. Insofern bringt der alternative Einsatz von JavaScript keine Vorteile.



Bedenklich für die Anwendung ist, dass Sie dem Nutzer die Kontrolle darüber lassen, was gespeichert wird. Die JavaScript-Codes sind im Quelltext der HTML-Seite sichtbar oder lassen sich mit ein wenig technischem Verständnis sichtbar machen. Es besteht zwar prinzipiell die Möglichkeit, mit PHP die Parameter in JavaScript dynamisch zu erzeugen und so die Cookies praktisch »fernzusteuern«, sinnvoll ist dies aber im Sinne halbwegs verständlich programmierter Skripte nicht.

4.4 Verbindungssteuerung

Bislang wurde jedes Skript einzeln aufgerufen; Nutzer, die gleichzeitig zugreifen, erhalten jeweils eine neue Version. Auch derselbe Nutzer wird immer wieder ein »neues Skript« erhalten, was unkritisch ist, solange der Inhalt sich sowieso nicht ändert. Um den Nutzer wiederzuerkennen, sind zusätzliche Anstrengungen notwendig.

4.4.1 Funktionsübersicht

Folgende Funktionen stehen zur Verfügung:

- `ignore_user_abort`
Verhindert, dass Nutzer mit dem ABBRUCH- (STOP)-Schalter die Verbindung unterbrechen.
- `register_shutdown_function(shutdown)`
Registriert eine Funktion *shutdown*, die nach einem Abbruch oder dem Ende des Skripts aufgerufen wird. Die Funktion darf keine Ausgaben enthalten.
- `connection_aborted`
Stellt fest, ob die Verbindung unterbrochen wurde.
- `connection_timeout`
Stellt fest, ob es zu einem Abbruch aufgrund einer Zeitüberschreitung kam.
- `connection_status`
Gibt alle Statuswerte zurück.
- `set_time_limit(seconds)`
Setzt die Zeitbegrenzung *seconds* in Sekunden.

4.4.2 Einführung

Das verbindungslose Protokoll HTTP führt bei der praktischen Umsetzung von Webprojekten zu einigen Problemen, denn der Status eines Clients kann von großer Bedeutung sein. Es gibt deshalb eine Reihe von Funktionen in PHP, die der Verbindungssteuerung (engl. *connection handling*) dienen.

Normal
Aborted
Timeout

Intern kennt PHP drei Zustände für Verbindungen:

- *Normal* (normal, Code 0)
- *Abgebrochen* (aborted, Code 1)
- *Zeitüberschreitung* (timeout, Code 2)

Während der Abarbeitung eines PHP-Skripts wird der Status *Normal* sein. Wenn der Client die Abarbeitung abbricht (durch Verlust der Verbindung oder Klick auf den ABBRECHEN-Schalter), wird der Status *Abgebrochen* gesetzt. Wird die Ausführung des Skripts nicht innerhalb der normalen Zeitbegrenzung beendet, unterbricht PHP die Ausführung und setzt den Status *Zeitüberschreitung*.

ignore_user_abort()
set_time_limit()

Die Funktion `set_time_limit` kann diesen Wert verändern, wenn der PHP-Interpreter nicht im sicheren Modus betrieben wird. Im sicheren

Modus – den viele Provider anwenden – können solche Systemwerte nicht durch Nutzerskripte geändert werden.

Es ist möglich, die Abarbeitung des Skripts fortzusetzen, auch wenn der Client die Unterbrechung wünscht. Dann erfolgt zwar trotzdem die Trennung und die gesendeten Daten werden im Browser nicht mehr angezeigt, bestimmte Prozesse können aber im Skript sauber beendet werden. Dies gilt vor allem für Datenbankverbindungen, die häufig explizit geschlossen werden müssen, um Ressourcen freizugeben. Setzen Sie dazu die folgende Option oder fügen Sie diese hinzu, wenn sie in der Datei PHP.INI noch nicht vorhanden ist:

```
ignore_user_abort = 0n;
```

Alternativ kann auch die Funktion `ignore_user_abort` genutzt werden:

```
$status = ignore_user_abort();  
ignore_user_abort(1);
```

Ohne Parameter wird der aktuelle Status zurückgegeben. Mit Parameter wird der aktuelle Wert gesetzt.

Mit der Funktion `register_shutdown_function` wird der durch den Nutzer provozierte Abbruch registriert. Sie können so beim nächsten Start des Skripts feststellen, ob vorher ein Abbruch erfolgte. Als Argument wird der Name einer Funktion übergeben, die im Falle eines Abbruchs ausgeführt wird.

Die Funktion darf keine Ausgaben mehr an den Browser senden. Der Befehl `echo` und die Funktion `print` sind nicht erlaubt, ebenso wie die Nutzung von `setcookie`.



Hinweis

Eine primitive Möglichkeit wäre das Registrieren eines Abbruchs in einer speziellen »Abbruchdatei«. Sie könnten eine Session-ID in eine Datei schreiben und beim nächsten Start des Skripts durch denselben Nutzer darauf reagieren. Darauf wird im [Abschnitt 4.5 Sessionverwaltung](#) ab Seite 336 noch detailliert eingegangen.

Sie können die Funktion `connection_aborted` abfragen. Wenn die Verbindung zum Client unterbrochen ist, wird `True` zurückgegeben, sonst `FALSE`. Die Verbindung wird durch physische Trennung der Leitungen, Abbruch durch den Nutzer oder Ablauf der Zeitbegrenzung (Standard: 30 Sekunden) unterbrochen.

Die Zeitbegrenzung können Sie mit der Funktion `set_time_limit` setzen. Die Funktion `connection_aborted` wird `TRUE` und eine zuvor registrierte Shutdown-Funktion wird ausgeführt. Damit Sie unterscheiden können, ob eine Zeitüberschreitung oder ein provozierter Abbruch erfolgten, testen Sie die Funktion `connection_timeout`. Wenn die Funktion `TRUE` zurückgibt, wurde die Verbindung durch Zeitüberschreitung unterbrochen.

php.ini

register_shutdown_function()

connection_aborted()

set_time_limit()

connection_timeout()

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E



Hinweis

Wenn Sie PHP bei einem Provider nutzen, wird der Interpreter normalerweise im sicheren Modus (safe mode) betrieben. Sie können dann nicht auf Systemfunktionen zugreifen und die Einstellung ändern, die in der Datei `PHP.INI` festgelegt wurden.

connection_status()

Die Zustände *Aborted* und *Timeout* können gleichzeitig auftreten. Das kann vorkommen, wenn Sie das Unterbrechen eines Skripts durch den Nutzer verboten haben (`ignore_user_aborts`), der Nutzer den Abbruch aber verlangt hat. Dann wird das Skript weiter ausgeführt, läuft möglicherweise bis zur Zeitüberschreitung und setzt dann beide Funktionen. Für die Überwachung eignet sich gut die Funktion `connection_status`, mit der alle drei Zustände gleichzeitig erfragt werden können. Die Darstellung erfolgt in Form eines Bitfeldes, wenn *Aborted* (1) und *Timeout* (2) aktiv sind, wird 3 zurückgegeben.

4.4.3 Ausführungssteuerung

Die Ausführung von Skripten lässt sich vielfältig beeinflussen. Für die Beendigung kennen Sie bereits die Befehle `die` und `exit`, die in ➡ Abschnitt 3.5.1 *Abbruchsteuerung* ab Seite 271 vorgestellt wurden.

sleep() usleep()

Die Ausführung kann jedoch nicht nur unterbrochen, sondern auch verzögert werden. Dazu setzen Sie die Funktionen `sleep` oder `usleep` ein. Der einzige Unterschied ist die Größenordnung des Parameters. `sleep` erwartet die Angabe in Sekunden, während `usleep` mit Mikrosekunden rechnet. Die beiden folgenden Angaben sind identisch:

```
sleep(3);
usleep(3000000);
```



Windows

Die Funktion `usleep` steht unter Windows nicht zur Verfügung, weil die Zeitmessung auf Betriebssystemfunktionen basiert. Windows stellt jedoch keinen Zeittakt mit der Genauigkeit Mikrosekunden bereit.

4.5 Sessionverwaltung

Um die im letzten Abschnitt angesprochenen Sitzungen (engl. *sessions*) zu kontrollieren, ist die so genannte Sitzungs- oder Sessionverwaltung notwendig. Dieser Abschnitt zeigt, wie mit PHP eine komfortable Sitzungsverwaltung möglich ist.

4.5.1 Einführung in die Sessionverwaltung



Theorie

Beim Umgang mit den ersten, etwas umfangreicheren Skripten wird schnell klar, welche gravierenden Nachteile HTTP hat. Der verbindungslose Ablauf ist zwar bei kleinen Skripten vorteilhaft, weil die Datenübertragung flexibel, störsicher und einfach abläuft. Wenn Sie

aber Skripte aus mehreren Formularen zusammensetzen, gibt es Schwierigkeiten. Nutzer können, wenn sie auf der folgenden Seite landen, nicht ohne weiteres wieder identifiziert werden. Der Server vergisst sämtliche Informationen, die er aus der letzten Transaktion ermittelt hat. Ob ein Nutzer eine Seite zweimal betritt oder ob es sich um zwei verschiedene Nutzer handelt, kann nicht sicher unterschieden werden.

Natürlich gibt es verschiedene Lösungen für die Problematik. Welche Lösung Sie wählen, hängt von der Aufgabenstellung und der zur Verfügung stehenden Technik ab. Letzteres ist mit PHP leicht zu entscheiden, denn fast alle Techniken werden gleichermaßen gut unterstützt. Einfluss auf die Entscheidung hat aber auch die Serverkonfiguration selbst. Bestimmte Eigenschaften des Apache können genutzt werden, andere jedoch nur, wenn PHP als Modul einkompiliert wurde. Steht kein Apache zur Verfügung, schränkt dies die Möglichkeiten etwas ein. Das Kapitel zeigt deshalb vorrangig solche Lösungen, die plattformunabhängig arbeiten. Auf die Besonderheiten des Apache wird außerdem eingegangen.

Der richtige Weg

Bei der Auswahl der Methode zur Sitzungsverfolgung können Sie grundsätzlich folgende Wege beschreiten:

- Speichern des Status in versteckten Feldern (hidden fields)
- Nutzung von Cookies
- Speichern des Status in der URI
- Nutzung des Prozessspeichers des Webservers (nur Apache)
- Speichern in einer Datei
- Speichern in einer Datenbank

Alle anderen Verfahren, wie das in Kapitel 12 vorgestellte Projekt *phpTemple*, basieren auf einer dieser Techniken. Man kann die Techniken zum besseren Verständnis noch in zwei Gruppen einteilen. Die Punkte 1 bis 3 arbeiten mit clientseitiger Unterstützung, 4 bis 6 dagegen basieren nur auf Abläufen im Server.

Jede Methode hat spezifische Vor- und Nachteile. Die clientseitigen Techniken verursachen kaum einen höheren Overhead bei der Datenübertragung und beanspruchen den Webserver nicht. Nachteilig ist die erforderliche Kooperation der Browser, die nicht immer sichergestellt werden kann. Abgesehen davon bieten sich mit der Auslagerung von Funktionen in den Browser auch umfangreiche Manipulationsmöglichkeiten. Viele erfolgreiche Site-Hacks basieren auf der Ausnutzung dieser Methoden. Vor allem versteckte Felder und URI-Daten

Vor- und Nachteile

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

sind ohne weiteres sichtbar (zumindest nach Wahl der Funktion ANSICHT | QUELLE). L. Stein und D. MacEachern berichten in ihrem Buch »Writing Apache Moduls« (O'REILLY), dass kommerzielle Shoppingsites feststellen, dass bei bis zu 30% der Zugriffe versucht wird, die sichtbaren Daten zu manipulieren, um Funktionen zu stören oder ungerechtfertigten Zugriff zu erlangen. Cookies sind nicht ganz so einfach zu manipulieren, werden aber von vielen Nutzern nicht akzeptiert und schränken die Einsatzbreite einer Site etwas ein. Die gleiche Quelle berichtet, dass etwa 20% der Browser Cookies nicht unterstützen, wobei nicht unterschieden wird zwischen Browsern, die Cookies technisch nicht ausführen oder Nutzern, welche die Ausführung unterbinden. Grundsätzlich kann man aber davon ausgehen, dass alle Browser Cookies (technisch) unterstützen.

Serverseitige Techniken sind unabhängig vom Browser und weniger anfällig für Angriffe. Speichert der Server die Sitzungsdaten im Speicher, sind erhebliche Zugriffe auf Systemressourcen zu beobachten. Auch wenn nur wenige Byte gespeichert werden, benötigt wird oft weitaus mehr Speicher. Bei Tausenden parallel ablaufenden Zugriffen können auch größere Maschinen aufgeben. Serverseitige Operationen sind komplexer und schwerer zu implementieren. So müssen Sie den Umgang mit abgerissenen Verbindungen beachten, die »Haltezeit« für eine Sitzung berechnen und gleichzeitig den Verbrauch an Speicher überwachen. Die Bedienung kann unter Umständen erschwert werden, denn der Browser selbst hat kein eindeutiges Merkmal. Alle übertragenen Daten, wie Browsername, Betriebssystem oder Version sind nicht eindeutig genug. Auch die IP-Nummer ist nicht geeignet, denn einige große Provider vergeben die IP nicht nur beim Verbindungsaufbau dynamisch, sondern wechseln sie auch während der Sitzung. Ohne die Unterstützung mit einer der clientseitigen Techniken bleibt nur die Hilfe des Nutzers – Loginname und Kennwort einzugeben.

Bei der Auswahl ist auch das Ziel zu beachten. Wenn Sie nur eine Verfolgung über ein paar Skripte benötigen, sind versteckte Felder und URI zu bevorzugen. Wollen Sie den Nutzer auch am nächsten Tag wiedererkennen, sind Cookies eine Variante. Wechseln die Nutzer häufig die Computer, kann nur eine datenbankbasierte Lösung helfen.

4.5.2 Sessions mit versteckten Feldern

Neben der bekannten Session-ID, die den Nutzer zweifelsfrei identifiziert, ist oft auch die Übertragung verschiedener Einstellungen nötig. Die Weiterreichung von Skript zu Skript ist eine sehr ressourcensparende Methode, denn sonst müssten für alle Nutzer die aktuellen Einstellungen auf dem Server gespeichert werden.

Einschränkungen

Versteckte Felder können nur genutzt werden, wenn die Bewegung des Nutzers von Skript zu Skript mit Formularen geschieht. Der einzige Grund, keine Formulare zu verwenden, liegt oft in den unerwünschten Schaltflächen. Es gibt jedoch keinen Grund, diese zu verwenden. Das folgende Tag ersetzt den Absende-Schalter:

```
<INPUT type="image" src="bilddatei.gif">
```

Das folgende Beispiel zeigt – noch ohne die Sessionfunktionen in PHP 4 – wie eine Session-ID im Bedarfsfall erzeugt und in einem versteckten Feld gespeichert wird.

```
<?php
if (!$session) {
    $seed = "SeedwertFuerSchluessel" . $REMOTE_ADDR . time();
    $session = md5($seed);
}
echo <<<FORM
<form action="$PHP_SELF" method="post">
<input type="hidden" value="$session" name="session" />
Session ID : $session<p />
<input type="submit" value="Weiter... " />
</form>
<a href="$PHP_SELF">Neustart</a>
FORM;
?>
```

*Listing 4.22:
gensessionid:
Erzeugen einer
Session-ID im
Bedarfsfall*

Wichtig ist, dass der Name des Tags name=session dem der Variablen entspricht. Auf der nächsten Seite wird dieselbe Sequenz erneut ausgeführt. Wenn ein verstecktes Feld übergeben wurde, erzeugt PHP automatisch eine Variable mit dem Namen und dem Inhalt des Feldes. Die Übergabe auf das nächste Skript erfolgt mit dem <FORM>-Tag.

Wie es funktioniert

Beachten Sie, dass versteckte Felder außerhalb des <FORM>-Containers wirkungslos bleiben.



Es ist sinnvoll, die auf jeder Seite erforderliche Abfrage der Informationen in einer Datei abzulegen, die mit include eingebunden wird. Wenn die Informationen auf einer der folgenden Seiten nicht zur Verfügung steht, sollte eine Umleitung auf die Startseite erfolgen.

Wenn sich alle Funktionen in ein und demselben Skript abspielen, können Sie die globale Variable \$PHP_SELF nutzen, um das Ziel des Formulars anzugeben. Zuerst die Schreibweise im HTML-Code:

\$PHP_SELF

```
<form action="<? echo $PHP_SELF; ?>" method="post">
```

Innerhalb eines PHP-Skripts sieht es folgendermaßen aus:

```
echo "<form action=\"$PHP_SELF\" method=\"post\">";
```

Im Beispiel wird dagegen die bequeme »Heredoc«-Syntax verwendet.

4.5.3 Sessions mit URI

Statt der Übertragung mittels HTTP-POST, wie in ➡ Abschnitt 4.5.2 *Sessions mit versteckten Feldern* ab Seite 338 beschrieben, kann auch der URL¹⁶ selbst erweitert und damit die GET-Methode genutzt werden. ➡ Abschnitt 4.2 *Daten per URL weiterreichen* ab Seite 318 zeigte bereits alle nötigen Techniken. Sie können also auch hier einfach jeden weiterführenden Link um entsprechende Informationen ergänzen.

Wie schon bei den versteckten Feldern liegen die so übertragenen Informationen recht offen dar. Sie können zwar mit Verschlüsselungstechniken und aufwändigen Zufallsgeneratoren einigen Schutz erreichen, Manipulationen sind und bleiben aber immer ein Risiko. Gegen den versierten Hacker ist der Schutz ohnehin recht eingeschränkt, zum Glück sind diese Leute selten. Viel häufiger werden Sie mit verspielten, aber fachlich unkundigen Nutzern konfrontiert. Die richten dann auch unwissentlich den größten Schaden an.

Damit Leute erst gar nicht auf die Idee kommen, nach Informationen im URI zu suchen, können Sie alles in einem unsichtbaren Frameset verstecken. Betrachten Sie die folgende Definition:

```
<frameset rows="100%,*">
```

Hier wird ein Frame definiert, das 100% des Browserfensters für sich in Anspruch nimmt, das andere Frame darf dagegen den »Rest« nutzen. Wenn Sie nun innerhalb des sichtbaren Frames (das unsichtbare wird nicht genutzt) Ihre Skripte ablaufen lassen, werden die Daten im Browser nicht sichtbar. Sie sind zwar weiter im Quelltext der HTML-Seite zu sehen, aber man wird nicht so offensichtlich drauf gestoßen.

4.5.4 Sessions mit Cookies

Die ausführliche Einführung in Cookies macht natürlich nur Sinn, wenn es auch praktische Anwendungen gibt. Fast jede Applikation kann davon profitieren. HTTP ist, wie in ➡ Abschnitt 2.2.1 *HTTP* ab Seite 72 bereits dargestellt, ein verbindungsloses Protokoll. Jeder Anforderung steht eine Antwort gegenüber, Response folgt auf Request. Dazwischen geht die Verbindung praktisch verloren. Der Server kann nicht erkennen, ob die nächste Anforderung für den gleichen Browser oder einen anderen bestimmt ist. Da IP-Nummern oft dynamisch vergeben werden und bei einigen Providern sogar während des Surfens wechseln, eignen sich die Verbindungsdaten nicht zur Zuordnung.

¹⁶ URI (uniform resource identifier) ist die allgemeinere Form des URL. Als URL wird im Allgemeinen nur die Adresse bezeichnet, während URI die komplette Zeichenkette bezeichnet.

Andererseits kann niemand verlangen, dass sich Nutzer Seite für Seite immer wieder neu anmelden.

Solche Abläufe, bei denen sich der Nutzer auf einer Site bewegt, werden als Sitzung (engl. *session*) bezeichnet. Eine der vordringlichsten Aufgaben der Cookies ist es, eine Session zu speichern und dem Server damit die Verfolgung des Nutzers zu ermöglichen. Dieser Abschnitt beschreibt aber auch »cookieleose« Alternativen zur Sessionsteuerung.

Sessions mit Cookies verwalten

Die einfachste Methode, Sessions zu verwalten, bieten Cookies. Dabei wird am Beginn der Session ein Cookie gespeichert, das kein Verfallsdatum enthält. Der gespeicherte Inhalt muss den Nutzer eindeutig identifizieren. Eine Kombination aus IP-Nummer und Zufallszahl hat sich dabei bewährt. Um die angezeigten Daten vor Manipulationen zu schützen, kann außerdem eine MD5-Verschlüsselung erfolgen. Eine »Rückübersetzung« ist dagegen nicht notwendig. Es geht ja nur um die Feststellung »Wer ist wer?«.

Auf der Startseite der Applikation wird die aktuelle Session-ID abgefragt – ist keine vorhanden, wird das entsprechende Cookie erzeugt und gespeichert. Alle folgenden Seiten fragen dies ebenfalls ab und identifizieren damit den Nutzer eindeutig.

```
<?php
if (!$sessionid) {
    mt_srand((double) microtime()*1000000);
    $sessionid = md5(str_replace(".", "", $REMOTE_ADDR) +
                    mt_rand(100000, 999999));
    setcookie("sessionid", $sessionid);
}
?>
<h3>Session-ID in Cookie speichern</h3>
<?php
echo "Es wurde folgende ID erzeugt: $sessionid";
?>
```

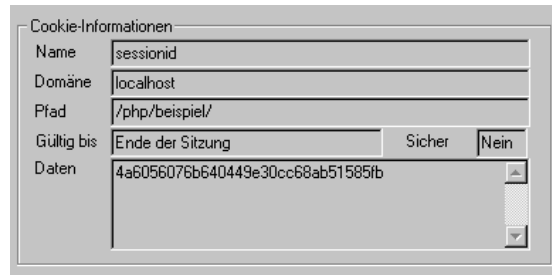
Das Beispiel in Listing 4.23 erzeugt die Session-ID aus der um die Punkte erleichterten IP-Adresse (62.208.3.6 wird also »6220836«), erhöht um einen Zufallswert zwischen 100 000 und 999 999. Diese Zahl wird mit dem MD5-Algorithmus kodiert. Daraus entstehen Hexzahlen-Kombinationen wie in Abbildung 4.4 gezeigt.

Die MD5-Verschlüsselung wird in RFC 1321 beschrieben

*Listing 4.23:
session_cookie: Erzeugen einer Session-ID und Ablage in einem Cookie*

Wie es funktioniert

Abbildung 4.4:
So sieht das Cookie
der Session-ID aus
Listing 4.23 aus



uniqid()

Sie können, wenn Sie die IP-Nummer nicht einbeziehen möchten, auch auf die Funktion `uniqid` zurückgreifen. Die Funktion nimmt die aktuelle Zeit in Mikrosekunden als Basis. Als Parameter kann ein Präfix angegeben werden, der den Zeitstempel verschiebt. Damit wird verhindert, dass auf mehreren parallel laufenden Computern identische IDs generiert werden. Unabhängig von dieser Vorgehensweise ist die Verschlüsselung mit MD5 auch hier anzuraten:

```
$sessionid = md5(uniqid(rand()));
```

4.5.5 Textdateien und Datenbanken

Aber normalerweise gehören zu einer Sitzung viele Daten, nicht nur die blanke Session-ID. Die zusätzlichen Daten können natürlich auch, wie bereits am Anfang des Kapitels gezeigt, mit GET und POST oder mit Cookies weitergereicht werden. Das ist aber nicht immer ratsam.

Textdateien

Sie könnten beispielsweise die Session-ID per URI oder Formular weitergeben, die zusätzlich nötigen Daten jedoch in einer gleichermaßen eindeutig benannten Datei speichern. Das Verfahren stellt allerdings für den Server eine verhältnismäßig hohe Belastung dar.

Datenbanken

Die Datenbanktechnik wird in den Kapiteln 6 bis 8 ausführlich behandelt. Beispiele zur Sitzungsverwaltung mit Datenbanken finden Sie beispielsweise im Projekt *phpTemple*. Dort werden Sitzungsdaten in einer MySQL-Datenbank abgelegt und sind vor Manipulationen sicher.

In allen Fällen ändert diese Vorgehensweise nichts am Prinzip mit dem Umgang mit Session-IDs. Die Skriptsteuerung wird nur für den Benutzer weniger transparent und damit schlechter angreifbar. Für große Projekte kann außerdem deutlich universeller und damit wartungsfreundlicher programmiert werden.

4.5.6 Sicherung der Client-Techniken

Im letzten Abschnitt wurde bereits die MD5-Verschlüsselung genutzt, um Daten so zu verpacken, dass eine Manipulation sinnlos erscheint. Der Sinn besteht im Wesentlichen darin, auf dem Computer des Nut-

zers gespeicherte Daten nicht zu einer wertvollen Angriffswaffe in den Händen eines Hackers werden zu lassen.

Stellen Sie sich folgendes Szenario vor: Sie betreiben eine Seite, die Produkte verkauft. Sie speichern die Kreditkarten Ihrer Kunden, damit diese ihre Daten nur einmalig über das Netz senden müssen. Um Nutzer während der Sitzung wiederzuerkennen, werden versteckte Felder genutzt. Wenn ein Nutzer nun seinen Arbeitsplatz verlässt und den Computer ausschaltet, kann der Server dies nicht erkennen. Er wird die Sitzung erst einige Zeit später »vergessen«. Ein anderer Mitarbeiter geht nun an den Computer, durchsucht das Verzeichnis mit den gespeicherten Seiten (Cache) und ruft die zuletzt genutzte Seite auf. Er wird damit als Nutzer wiedererkannt und kann die letzte Sitzung unmittelbar fortsetzen. Vielleicht kann er dann auf die Kreditkartendaten zugreifen, die auf einer Serviceseite geändert werden können.

Zur Sicherung müssen Sie Sitzungen mit einem engen Zeitraster belegen. Nach einer bestimmten inaktiven Zeit sollten Sitzungen ungültig werden, der Nutzer muss sich erneut anmelden. Diese Technik nutzen auch Banken, die Deutsche Bank beispielsweise erzwingt nach 10 Minuten Inaktivität die erneute Eingabe der PIN.

Es macht also Sinn, innerhalb der Übertragung der Nutzerdaten auch die ursprüngliche Zeit zu verstecken. Und um nicht so offen dieses Verfahren zu propagieren, wird ein weiterer MD5-Code erzeugt.

Nehmen Sie diesen Code nicht allein als Sitzungsnummer, denn MD5 ist kein Zufallsgenerator; gleiche Zeiten erzeugen auch gleiche Codes, sodass viele Nutzer denselben Zeitstempel haben können!



Die Vorgehensweise ist recht einfach, man muss eben nur daran denken:

```
<INPUT type=hidden value=<?php echo md5(time); ?> name=tst>
```

Sitzungen zeitlich begrenzen

Aus Sicherheitsgründen kann es oft erforderlich sein, Sitzungen nach einer bestimmten Zeit der Inaktivität zu unterbrechen und die erneute Eingabe der Anmeldedaten zu verlangen. Das folgende Skript realisiert eine solche Vorgehensweise.

```
<?php
$timeout = 5; // seconds
if (!isset($first)) $first = 1;
$nextid = time();
if ((( $nextid - $id ) > $timeout) & !$first) {
    $diff = $nextid - $id;
    echo <<<OUT
        Timeout<br>
```

Listing 4.24:
session_timeout:
Sessionverwaltung
mit Timeout (5
Sekunden nur zu
Testzwecken)


```

        Die Session war mehr als <b>$diff</b> Sekunden inaktiv.
        <a href="$PHP_SELF?script=session_timeout">
        Neustart der Session</a>
    OUT;
    exit;
} else {
    echo "Session läuft!";
    $first = 0;
    $id = $nextid;
}
echo <<<FOOTER
    <p>
    <a href="$PHP_SELF?id=$id&first=$first">Restart</a>
FOOTER;
?>

```

Wie es funktioniert Zur Berechnung der Dauer einer Session wird (zusätzlich zur Session-ID, die hier nicht explizit ausgeführt wurde) ein Zeitstempel mitgeführt. Mit der folgenden Anweisung wird die Dauer der maximalen Zeit der Inaktivität in Minuten festgelegt:

```
$timeout = 10;
```

Beim ersten Start des Skripts muss der Anfangszustand erkannt werden:

```
if (!isset($first)) $first = 1;
```

Bei jedem Durchlauf wird die momentane Zeit ermittelt:

```
$nextid = time();
```

Nun wird diese Zeit so umgerechnet, dass sich die resultierenden Werte erst ändern, wenn die Zeit überschritten wurde. Alter und neuer Wert werden verglichen. Der Vergleich wird beim ersten Durchlauf des Skripts unterdrückt. Ist die Zeit überschritten, wird das Skript beendet (exit):

```
if (($nextid - $id) > $timeout) & !$first) {
```

Beim ersten Durchlauf oder innerhalb der aktiven Zeit wird der aktuelle Zeitstempel weitergesetzt:

```

    $first = 0;
    $id = $nextid;

```

Der HTML-Teil zeigt, wie das Skript sich selbst aufruft und die Daten übergibt:

```
<a href="$PHP_SELF?id=$id&first=$first">Restart</a>
```

4.5.7 PHP-Sessionfunktionen

Die Sessionfunktionalität wurde erst in PHP 4 hinzugefügt. Bis dahin war ein komfortabler Umgang mit Sessions nur über Zusatzprogramme möglich.



Die in PHP verwendeten Sessionfunktionen basieren auf Cookies oder – alternativ – auf der Weitergabe der Sessioninformationen über die URL, erlauben jedoch eine ungleich komfortablere Programmierung.

Funktionsübersicht

- `session_start`
Initialisiert eine Session und muss auf jeder Seite am Anfang ausgeführt werden.
- `session_destroy`
Zerstört alle Sessiondaten. Der nächste Aufruf von `session_start` startet eine neue Session.
- `session_name`
Setzt oder holt den Namen der aktuellen Session.
- `session_module_name`
Setzt oder holt das aktuelle Session-Modul. Außer dem eingebauten Modul kann auch ein selbstgeschriebenes verwendet werden.
- `session_save_path`
Setzt oder holt den abgesicherten Pfad der aktuellen Session. Dies gilt nur, wenn die Daten in Dateien gespeichert werden.
- `session_id`
Setzt oder holt die aktuelle Session-ID. Soll normalerweise nur gelesen werden.
- `session_register`
Bindet eine Variable an die aktuelle Session. Die Variable steht damit auf der nächsten Seite zur Verfügung.
- `session_unregister`
Löst eine Variable aus der aktuellen Session.
- `session_is_registered`
Ermittelt, ob eine Variable an die aktuelle Session gebunden wurde.

- `session_get_cookie_params`

Parameter des Session-Cookies holen.

- `session_set_cookie_params`

Parameter des Session-Cookies setzen.

- `session_decode`

Dekodiert Sessiondaten aus einer Zeichenkette. Normalerweise erfolgt dies automatisch und muss nicht selbst erledigt werden.

- `session_encode`

Kodiert Sessiondaten in einer Zeichenkette. Normalerweise erfolgt dies automatisch und muss nicht selbst erledigt werden.

Einführung in PHP-Sessions

PHP-Sessions basieren auf der Vergabe einer eindeutigen Session-ID. Die Verknüpfung zwischen den Zugriffen kann entweder über Cookies oder über die URL erfolgen. Wie diese Mechanismen prinzipiell arbeiten, wurde bereits im vorangegangenen Abschnitt erläutert. Durch spezielle Funktionen können auch Variablen an die Session gebunden werden. Dies erleichtert den Aufbau entsprechender URLs bzw. die Übergabe der Werte an Cookies. Sessions können ständig aktiviert sein, in diesem Fall ist `session.auto_start` in der Konfigurationsdatei `PHP.INI` auf 1 (TRUE) zu setzen. Andernfalls löst eine der Funktionen `session_start` die Anforderung der Sessionwerte explizit oder, bei `session_request`, implizit aus. Bei der Speicherung von Variablen ist zu beachten, dass diese automatisch serialisiert werden. Unter dem Serialisieren ist zu verstehen, dass Variablen in eine einfach speicherbare Form gebracht werden, das heißt, in eine Zeichenkette. Andernfalls könnte man Arrays nicht in Cookies speichern.



Hinweis

Objekte lassen sich derzeit mit PHP-Sessions nicht speichern. Verwenden Sie für komplexe Daten deshalb Arrays.

Das folgende Beispiel zeigt die Registrierung einer Variablen und die Übertragung auf die nächste Seite. Beachten Sie, dass die Session selbst implizit initialisiert wird:

Listing 4.25:
sessiontest: Sessions
einfach verwenden
(Cookies erforderlich)

```
<?php
session_register("counter");
$counter++;
?>
<h3>Counter</h3>
<?php
$sessionid = session_id();
echo <<<OUT
    Hallo, Sie waren schon <b>$counter</b> mal auf dieser Seite!
```

```
<p />
  Machen Sie bitte <a href="$PHP_SELF?">hier</A> weiter.
OUT;
?>
```

Möglicherweise funktioniert dies nicht auf Ihrem Computer, weil die Sessions nicht korrekt konfiguriert sind. Dies wird im nächsten Abschnitt beschrieben.

Sessions konfigurieren

Wenn Sie auf einem Server nur eine Applikation laufen haben, ist es sinnvoller, bestimmte Einstellungen in der Konfigurationsdatei `PHP.INI` vorzunehmen. Die folgenden Einträge können Sie verwenden:

```
session.save_handler = dateihandle;
```

Diese Option setzt ein Handle auf eine Datei, die zum Speichern der mit der Session verbundenen Daten verwendet wird. Der Standardwert ist `files`, ein internes Handle.

```
session.save_path = argument;
```

Mit diesem Eintrag können Sie zusätzliche Argumente an die Datei übermitteln. Der Standardwert ist `/TMP`.

Der Pfad `/TMP` steht unter Windows nicht zur Verfügung. Entweder legen Sie ein entsprechendes Verzeichnis mit Schreib- und Leserechten an oder ändern den Pfad in der Konfigurationsdatei `PHP.INI`.



Die folgenden Optionen sind für alle Betriebssysteme identisch.

```
session.name = PHPSESSION;
```

Die definiert den Standardnamen der Session, der auch als Name des Cookies verwendet wird.

```
session.auto_start = 0;
```

Der Wert muss 1 sein, wenn bei jeder Anforderung das Session-Modul starten soll. Der Standardwert ist 0.

```
session.lifetime = 0;
```

Hiermit legen Sie die Lebenszeit der Session-Cookies fest. Der Standardwert ist 0 und bedeutet »bis der Browser geschlossen wird«.

```
session.serialized_handler = php;
```

Dieser Eintrag legt fest, wie Variablen serialisiert werden. Derzeit kann dort nur »php« stehen, die internen Serialisierungsfunktionen werden dann verwendet.

```
session.gc_probability = 1;
```

Die interne Routine zum Aufräumen des Speichers (gc = garbage collection¹⁷) kann nach einer bestimmten Anzahl Anforderungen gestartet werden. Die Angabe erfolgt in Prozent, der Standardwert ist 1.

```
session.gc_maxlifetime = 600;
```

Diese Option definiert die Anzahl der Sekunden, nach der Variablen aus Sessions als »Müll« angesehen werden und von der Aufräumroutine (engl. *garbage collection*) eliminiert werden.

Sessions einrichten und nutzen

session_start()
session_destroy()

Eine Session wird initialisiert, indem die folgende Funktion aufgerufen wird:

```
session_start();
```

Am Ende der Session können die Werte explizit zerstört werden:

```
session_destroy();
```

Beide Funktionen benötigen keine Parameter und geben immer TRUE zurück.

session_name()

Der Name einer Session, zugleich Name des Session-Cookies oder der GET-Variablen, kann folgendermaßen verändert werden:

```
session_name("name");
```

Der Standardwert ist PHPSESSID. Die Funktion gibt den aktuellen Namen auch wieder zurück.

```
$name = session_name();
```

Das folgende Beispiel zeigt die Anwendung:

Listing 4.26:
sessionname:
Name der Session
ändern (Ausschnitt)

```
<?php
$session = 'Anwendung 3';
session_name($session);echo "Der Sessionname ist " .
# Die Verwendung entspricht Listing 4.25 von Seite 346
echo "Die Session hat den Namen : " . session_name();
?>
```

Die Notwendigkeit, den Namen zu ändern, besteht unter normalen Umständen nicht. Wenn Sie jedoch eine Sessionsteuerung selbst entwerfen und dafür eine Datenbank verwenden, werden möglicherweise mehrere Sites Sessiondaten in derselben Tabelle ablegen. Dann

¹⁷ Der Begriff »garbage collection«, dt. etwa »Müll einsammeln«, bedeutet in der Programmierung ein Freigeben von Speicher, der durch nicht mehr benötigte Variablen belegt wird. Viele Programmiersprachen geben Speicher nicht sofort wieder frei, um die Leistung zu steigern.

kann einer Unterscheidung sinnvoll sein. Derartige Techniken werden verwendet, um Hochleistungsanwendungen zu programmieren. Dabei werden statt einem PHP-Server mehrere eingesetzt. Ein Netzwerklastverteilungssystem verteilt dann die Anfragen auf die verschiedenen Maschinen. Damit die Session nicht beim Wechsel von einer Maschine zur anderen verloren geht, arbeitet im Hintergrund ein gemeinsamer Datenbankserver. Da solche Systeme jedoch oft mehrere Sites betreiben, müssen diese wiederum unterschieden werden.

PHP-Sessions intern

In den vorangegangenen Abschnitten wurden verschiedene Techniken zur Speicherung von Daten vorgestellt. PHP nutzt das Dateisystem, um Sessiondaten zu speichern. Dabei werden sehr viele Dateien angelegt, für jede aktive Session eine. Sie sollten bei der Wahl des Dateisystems darauf achten, dass dieses mit sehr vielen kleinen Dateien effizient umgehen kann. Unter Linux sollten Sie *reiserfs* anstatt *ext2fs* verwenden. Wenn Ihr Betriebssystem die Möglichkeit bietet, die Clustergröße einzustellen, sollten Sie einen möglichst kleinen Wert einstellen, beispielsweise 1 KByte.

```
session_save_path("pfad zu den dateien");
```

Die Funktion `session_save_path` setzt den Pfad fest, unter dem die Sessiondaten gespeichert werden. Den aktuellen Wert können Sie mit der folgenden Anweisung ermitteln:

```
$sessionpath = session_save_path();
```

Der Parameter ist optional. Standardmäßig ist in `PHP.INI` der Name `/TMP` eingestellt.

Der Pfad `/TMP` steht unter Windows nicht zur Verfügung. Entweder legen Sie ein entsprechendes Verzeichnis mit Schreib- und Leserechten an oder ändern den Pfad in der Konfigurationsdatei `PHP.INI`.



Insider

`session_save_path()`



Hinweis Windows

Sessionvariablen

Ein wichtiger Aspekt jeder Sessionverwaltung sind die Sessionvariablen. Vor allem Umsteigern von ASP macht das Fehlen eines derartigen Features viel Kopfzerbrechen, wenn Anwendungen nach PHP transportiert werden sollen.

Sessionvariablen sind Variablen, die fest an eine Session gebunden werden. Sicher haben Sie schon damit gearbeitet, jede Variable, die von einer Seite zur nächsten mit HTTP-GET oder Cookies transportiert wird, ist eine Sessionvariable. Der Wert und der Name der Variablen ist an die Session gebunden. Ist die Session zu Ende, verfällt die Variable.

session_register()
session
_unregister()
session
_is_registered()

PHP 4 bietet mit der Sessionverwaltung nun nicht nur eine einfache Unterstützung, sondern gleich drei Funktionen zum komfortablen Umgang mit Sessionvariablen an. Um eine Variable für eine Session zu erzeugen, schreiben Sie:

```
$ok = session_register("variablenname");
```

Der Variablenname muss eine existierende globale Variable sein. Die Funktion gibt FALSE zurück, wenn der Vorgang misslang, sonst TRUE. Die Auswertung des Rückgabeparameters ist nicht zwingend notwendig. Manchmal kann es erforderlich sein, die Sessionvariable wieder zu löschen:

```
session_unregister("variablenname");
```

Um zu erkennen, ob eine bestimmte Variable als Sessionvariable gebunden wurde, nutzen Sie folgende Funktion:

```
$isregistered = session_is_registered("variablenname");
```

Die Funktion gibt TRUE oder FALSE zurück. Der Inhalt der so übermittelten Variablen wird in ihr selbst zurückgegeben, das heißt, die Variable steht automatisch auf der nächsten Seite zur Verfügung. Ebenso wird die Variable automatisch serialisiert (in eine Zeichenkette verwandelt), wenn es sich um ein Array handelt.

session_decode()
session_encode()

Um die Variablen wieder zurückzuholen, müssen die serialisierten Werte dekodiert werden. Dazu rufen Sie die folgende Funktion auf:

```
session_decode();
```

Entsprechend werden Variablen mit dem nachfolgenden Funktionsaufruf serialisiert, wenn Sie es explizit machen wollen:

```
session_encode("variable");
```



Hinweis

Mehr Informationen zu Sessions finden Sie in Abschnitt 11.6.2 *Sessionverwaltung* ab Seite 810, wo eine professionelle Sessionverwaltung auf Datenbankbasis vorgestellt wird.

4.6 Zugriff auf das Dateisystem

Mit PHP können Sie komfortabel auf das Dateisystem zugreifen. Damit ergeben sich umfangreiche Anwendungsmöglichkeiten, auch wenn keine Datenbank zur Verfügung steht.

4.6.1 Einführung

Der Zugriff auf das Dateisystem ist eine enorme Erleichterung bei der Umsetzung vieler Projekte. Sie können mit Hilfe von Dateien Daten dauerhaft speichern, Grundeinstellungen und Stammdaten lesen

(➡ Abschnitt 4.6.2 *Dateien lesen und schreiben* ab Seite 356) und damit sogar eine Datenbank ersetzen. Da Zugriffe auf das Dateisystem des Servers eine Sicherheitslücke darstellen, sind zusätzliche Kenntnisse über die Funktionen und mögliche Gegenmaßnahmen notwendig (➡ Abschnitt 4.8 *Sicherheit* ab Seite 400). Auch wenn Sie Nutzern keinen direkten Zugriff geben möchten, gibt es sinnvolle Anwendungen:

- *Logdateien schreiben*

Protokollieren Sie Nutzerbewegungen in eigenen Dateien.

- *Formulardaten ablegen*

Speichern Sie Formulare als Dateien ab.

- *Tipp des Tages*

Bauen Sie dynamische Daten in Ihre Websites ein, die andere als Textdatei abgelegt haben.

Die Arbeit mit Dateien erfolgt nach einem einfachen Schema. Zuerst wird die Datei geöffnet, dann werden Daten herausgelesen oder hineingeschrieben und dann wird die Datei wieder geschlossen. Wenn eine Datei nicht existiert, kann sie beim Öffnen automatisch angelegt werden.

Wenn eine Datei geöffnet ist, kann der gesamte Inhalt in eine Variable transportiert werden. Besser ist die Verwendung eines Dateizeigers. Dieser markiert eine bestimmte Position in der Datei und steuert die Ausgabe der enthaltenen Daten.

PHP kann zwischen Binär- und Textdateien unterscheiden. Textdateien werden im ASCII-Modus geöffnet und es bieten sich spezielle Bearbeitungsfunktionen dafür an. **Binär oder Text?**

Um mit Dateien arbeiten zu können, werden so genannte Datei-IDs oder Handles erzeugt. Ein Handle ist ein Verweis auf eine Datei. Funktionen zur Dateimanipulation nutzen Handles¹⁸ zur Zugriffssteuerung. Dies ist einfacher, als immer wieder den kompletten Pfad zu einer Datei anzugeben. **Handles**

Als Dateiname kann, außer einem physischen Pfad auf einem Unix- oder NT-System, auch eine HTTP- oder FTP-Adresse genutzt werden. Dann wird eine Verbindung zu einem Webserver (*http:*) oder einem FTP-Server aufgebaut. FTP-Verbindungen können lesen und schreiben, HTTP-Verbindungen können natürlich nur lesen. Mehr **http und ftp**

¹⁸ Im Buch wird sehr oft der Begriff »Handles« genutzt. Generell handelt es sich dabei um eine Referenz (Verweis) auf eine bestimmte Verbindung. Der Begriff ist üblich und wurde deshalb nicht übersetzt.

dazu finden Sie in ➡ Abschnitt 4.7 *Verbindungen zu Servern im Internet* ab Seite 378.

Funktionsübersicht

Die folgende Liste enthält alle Funktionen, die im Zusammenhang mit dem Dateisystem verwendet werden können:

- `basename`

Die Funktion extrahiert den Namen einer Datei aus einer vollständigen Pfadangabe.

- `chgrp`

Wechselt die Rechte an einer Datei für eine bestimmte Benutzergruppe. Diese Funktion ist nur für Unix verwendbar.

- `chmod`

Ändert die Rechte an einer Datei für den aktuellen Benutzer. Diese Funktion ist unter Windows nur eingeschränkt verwendbar.

- `chown`

Wechselt den Eigentümer der Datei. Auch diese Funktion kann nur unter Unix eingesetzt werden.

- `clearstatcache`

Löscht den Status-Cache. Viele Funktionen ermitteln Daten über Dateien und legen diese in einem Speicher in PHP ab, sodass spätere Zugriffe schneller verlaufen. Damit die Daten aktualisiert werden, muss der Status-Cache gelöscht werden.

- `copy`

Kopiert eine Datei.

- `dirname`

Extrahiert den Verzeichnisnamen aus einer vollständigen Pfadangabe. Diese Funktion ist nicht »intelligent«, das heißt, sie extrahiert einfach den letzten Teil einer Zeichenkette hinter dem schließenden Schrägstrich. Es spielt keine Rolle, ob ein derartiger Pfad tatsächlich existiert.

- `diskfreespace`

Gibt den freien Speicherplatz eines Laufwerks an.

- `fclose`

Schließt eine Datei.

- `feof`
Testet, ob der Dateizeiger am Ende der Datei steht (End of File).
- `fgetc`
Holt ein Zeichen von der Position des Dateizeigers.
- `fgetcsv`
Holt eine Zeile und untersucht nach kommaseparierten Feldern (CSV-Datei).
- `fgets`
Holt eine Zeile aus einer Textdatei.
- `fgetss`
Holt eine Zeile und analysiert HTML-Tags.
- `file`
Liest die gesamte Datei in ein Array ein. Als Trennzeichen werden die Zeilenumbrüche verwendet, die dabei erhalten bleiben.
- `file_exists`
Prüft, ob eine Datei existiert.
- `fileatime`
Gibt das Datum des letzten Zugriffs auf die Datei an.
- `filectime`
Gibt das Datum der Datei an.
- `filegroup`
Holt die Benutzergruppe, die die Datei besitzt. Diese Funktion gilt nur für Unix.
- `fileinode`
Holt eine inode-Information über eine Datei. Diese Funktion gilt nur für Unix.
- `filemtime`
Gibt das Datum zurück, an dem die Datei das letzte Mal geändert wurde.
- `fileowner`
Ermittelt den Namen des Besitzers der Datei.

- `fileperms`
Holt die Dateiattribute.
- `filesize`
Gibt die Größe der Datei in Bytes zurück.
- `filetype`
Gibt den Dateityp als Zeichenkette zurück.
- `flock`
Verriegelt die Datei gegen weitere Zugriffe. Diese Funktion arbeitet unter Windows nicht zuverlässig.
- `fopen`
Öffnet eine Datei oder URL. Das von dieser Funktion zurückgegebene Handle wird von anderen Funktionen verwendet, um auf die Datei zuzugreifen.
- `fpassthru`
Liest Daten beginnend am aktuellen Dateizeiger aus einer Datei und sendet sie direkt zum Browser.
- `fputs`
Schreibt Daten an die Stelle des Dateizeigers in eine Datei.
- `fread`
Liest Binärdaten aus einer Datei.
- `fseek`
Setzt den Dateizeiger auf eine bestimmte Position.
- `ftell`
Gibt die aktuelle Position des Dateizeigers an.
- `fwrite`
Schreibt Binärdaten in eine Datei.
- `set_file_buffer`
Stellt den Puffer für einen Dateizeiger ein.
- `is_dir`
Ermittelt, ob der Name ein Verzeichnis ist.
- `is_executable`
Teilt mit, ob die Datei ausführbar ist.

- `is_file`
Ermittelt, ob es sich um eine reguläre Datei handelt.
- `is_link`
Ermittelt, ob es sich um einen Link (Verweis) handelt.
- `is_readable`
Ermittelt, ob die Datei lesbar ist.
- `is_writable`
Ermittelt, ob in die Datei geschrieben werden kann.
- `link`
Erzeugt einen Verweis (Link). Die Funktion arbeitet nur für Unix bzw. Linux.
- `linkinfo`
Holt Informationen über einen Verweis (Link). Diese Funktion arbeitet nur für Unix/Linux.
- `pclose`
Schließt einen Prozess-Dateizeiger.
- `popen`
Öffnet einen Prozess-Dateizeiger.
- `readfile`
Liest eine Datei und sendet sie an den Browser.
- `readlink`
Gibt das Ziel eines Verweises zurück.
- `rename`
Benennt eine Datei um.
- `rmdir`
Entfernt ein Verzeichnis.
- `stat`
Gibt Informationen über eine Datei zurück.
- `lstat`
Gibt Informationen über einen Verweis zurück (nur für Unix bzw. Linux).

- `symlink`

Erzeugt einen Verweis (nur für Unix/Linux).

- `tempname`

Erzeugt einen eindeutigen, einmaligen Namen. Das Verfahren ist unter Windows und Linux verschieden, prinzipiell funktioniert es jedoch unter beiden Betriebssystemen.

- `touch`

Setzt das Datum der letzten Veränderung.

- `umask`

Änderung der aktuellem UMASK (nur für Unix/Linux).

- `unlink`

Löscht einen Verweis oder eine Datei.

Diese Liste enthält nur die Funktionen, die mit Dateien umgehen oder im Zusammenhang mit Dateien benötigt werden. Funktionen für die Bearbeitung von Verzeichnissen finden Sie am Anfang von ➔ Abschnitt 4.6.4 *Umgang mit Verzeichnissen* ab Seite 367.

4.6.2 Dateien lesen und schreiben

Dateien lesen und schreiben gehört zu den elementaren Vorgängen. Dieser Abschnitt beschreibt, wie die entsprechenden Funktionen eingesetzt werden.

Funktionsübersicht Dateien lesen und schreiben

Zum Schreiben und Lesen verwenden Sie folgende Funktionen:

- `readfile, file`
- `fopen`
- `fclose`
- `fgets`

Die Dateifunktionen im Detail

**`readfile()`
`file()`**

Die Funktion `readfile` liest eine Datei und sendet den gesamten Inhalt ohne weitere Bearbeitung an die Standardausgabe – dies ist normalerweise der Browser.

```
<?php
$file = "news.txt";
echo "Ausgabe der Datei $file:<br>"
readfile($file);
?>
```

Listing 4.27:
file_readfile: Ausgabe
einer Datei zum
Browser

Analog funktioniert auch `file`, die gelesene Datei wird aber in einem Array abgelegt. Jede Zeile wird zu einem Element eines eindimensionalen Arrays. Der Zeilenumbruch bleibt am Ende des Elements. Das folgende Beispiel liest eine Datei ein und gibt sie zusammen mit Zeilennummern wieder aus:

```
<?php
$file = "news2.txt";
echo "Ausgabe der Datei $file:<br>";
$filearray = file($file);
foreach($filearray as $num => $line) {
    printf ("%02d : %s<br>", $num, $line);
}
?>
```

Listing 4.28:
file_filenum: Datei
mit Zeilennummern

Die wichtigste Funktion zum Umgang mit Dateien ist `fopen`. Sie können den Dateinamen und ein Attribut angeben:

fopen()

```
$handle = fopen("dateiname", "attribut");
```

Das Attribut bestimmt, wie die Datei geöffnet wird:

**ASCII-Dateien
Attribute**

- **r**
Öffnet eine Datei zum Lesen und setzt den Dateizeiger auf den Anfang (das erste Zeichen in der Datei).
- **r+**
Öffnet eine Datei zum Lesen und Schreiben und setzt den Dateizeiger auf den Anfang (das erste Zeichen in der Datei).
- **w**
Öffnet eine Datei zum Schreiben. Wenn die Datei nicht existiert, wird sie angelegt. Wenn die Datei vorhanden ist und Daten enthält, werden diese gelöscht und die Länge wird auf 0 gesetzt.
- **w+**
Öffnet eine Datei zum Schreiben und Lesen. Wenn die Datei nicht existiert, wird sie angelegt. Wenn die Datei vorhanden ist und Daten enthält, werden diese gelöscht und die Länge wird auf 0 gesetzt.
- **a**
Öffnet die Datei zum Schreiben. Wenn die Datei nicht existiert, wird sie neu angelegt. Vorhandene Daten bleiben erhalten. Der

Dateizeiger steht am Ende der Datei (a steht für append, dt. anhängen).

- a+

Öffnet die Datei zum Schreiben und Lesen. Wenn die Datei nicht existiert, wird sie neu angelegt. Vorhandene Daten bleiben erhalten. Der Dateizeiger steht am Ende der Datei.

Binärdateien Attribute

Jedes der Attribute kann mit einem vorangestellten b kombiniert werden, um den Zugriff auf Binärdateien zu ermöglichen. Daraus ergeben sich dann folgende Kombinationen:

- br, br+
- bw, bw+
- ba, ba+

fclose()

Geöffnete Dateien müssen wieder geschlossen werden, um Systemressourcen zu schonen und anderen Prozessen den Zugriff zu ermöglichen.

```
fclose($handle);
```

fgets() fputs() fwrite()

Die Funktion fgets liest aus einer Datei; die Funktion fputs dagegen schreibt in die Datei. fgets liest von der Position des Dateizeigers, bis eines der drei folgenden Ereignisse eintritt:

- Die Anzahl Bytes (*laenge*), die angegeben wurde, ist erreicht
- Ein Zeilenende wurde erreicht
- Das Dateiende wurde erreicht

```
$var = fgets($handle, $laenge);
```

```
fputs($handle, $var, [$laenge]);
```

Die Funktion fwrite ist lediglich ein Alias für fputs, es gibt keine Unterschiede. Die Angabe der Länge *laenge* ist optional, wird sie angegeben, schreibt die Funktion nur diese Anzahl Bytes.

fread()

Die Funktion fread entspricht fast der Funktion fgets, interpretiert aber nicht den Zeilenumbruch. Damit wird tatsächlich bis zum Dateiende oder die angegebene Anzahl Bytes gelesen – die Funktion kann daher auch für Binärdateien genutzt werden:

```
$bytestream = fread($handle, 4096);
```

Das Beispiel liest 4 KByte aus einer Datei, die vom Handle *\$handle* adressiert wird.

Einige Beispiele für die Dateifunktionen

Das folgende Beispiel liest eine Datei zeilenweise und gibt nur die Zeile aus, die zu einem bestimmten Tag passt.

```
<?php
$select = date("D", time());
$fp = fopen("news2.txt", "r");
while ($line = fgets($fp, 1000)) {
    if (preg_match("/^\[".$select."\]+\]/", $line)) {
        echo fgets($fp, 1000) . "<br>";
    }
}
fclose($fp);
?>
```

Listing 4.29:
file_fgets:
Zeilenweises Lesen
einer Datei

Das Beispiel in Listing 4.29 enthält keine Besonderheiten, wendet aber mehrere übliche Techniken an, die Sie beherrschen sollten. Die Zuweisung liest die nächste Zeile aus der angegebenen Datei:

```
$line = fgets($fp,1000)
```

Der gesamte Ausdruck wird FALSE, wenn der Zugriff misslingt. Dies ist normalerweise nur am Ende der Datei der Fall. Hier werden also Zugriff, Auslesen der Daten und Dateiendeprüfung in einem einzigen Ausdruck zusammengefasst. Der Dateizeiger wird automatisch weiter gesetzt, sodass die Ausgabe mit die nächste Zeile ausgibt:

```
echo fgets($fp,1000). "<BR>"
```

Die komplette Nachrichtendatei hat den folgenden Aufbau:

```
[Mon]
Heute beginnt die Woche!
[Tue]
Nur noch 4 Tage
[Wed]
Das Wochenende naht
[Thu]
Fast fertig
[Fri]
Freitag nach 1...
[Sat]
Was suchst Du hier heute?
[Sun]
Was suchst Du hier heute?
```

Listing 4.30:
Nachrichtendatei für
das Skript aus
Listing 4.29

Die Auswahl nutzt einen einfachen regulären Ausdruck zur Prüfung, der den Wochentagsnamen und die eckigen Klammern am Zeilenanfang erkennt.

Das Schreiben in eine Datei ist auch nicht komplizierter. Das folgende Beispiel schreibt das aktuelle Datum und die Uhrzeit sowie die IP-Nummer des Clients in eine Datei und gibt die Datei anschließend

aus. Drücken Sie immer wieder auf Aktualisieren (Reload) im Browser, können Sie die Datei »wachsen« sehen.

Listing 4.31:
*file fputs: Eine
primitive
Protokolldatei*

```
<?php
$logfile = "logdata/logfile.log";
$fp = fopen($logfile, "a");
$logline = sprintf("%s, %s\n", date("d.M.Y h:m:s"), $REMOTE_ADDR);
fputs($fp, $logline);
fclose($fp);
echo 'Das ist der Inhalt der Log-Datei:<p />';
echo '<pre>';
readfile($logfile);
echo '</pre>';
?>
```



Hinweis

Das Beispiel in Listing 4.31 funktioniert nur, wenn der Webnutzer Schreibrechte im Unterverzeichnis /LOGDATA hat. Wenn Sie die CD einfach kopiert haben, müssen Sie die Verzeichnisrechte gesondert einstellen.

Die Anzeige der Zeilenanzahl ist leicht möglich, wenn die Funktion `file` in Verbindung mit `count` genutzt wird.

Listing 4.32:
*file file: Anzeige der
Anzahl Zeilen einer
Textdatei*

```
<?php
echo 'Die Log-Datei hat ';
echo count(file("logdata/logfile.log"));
echo ' Zeilen.';
?>
```

Manchmal wird der Inhalt einer Datei nicht in einem Array, sondern in einer einzigen Zeichenkette benötigt. Vor allem im Hinblick auf die regulären Ausdrücke bieten sich gute Verarbeitungsmöglichkeiten an. Einen direkten Befehl gibt es nicht, aber eine einfache Befehlskombination:

```
$superstring = implode("", @file("datei.txt"));
```

Zur Anwendung kommen die Funktionen `file` (holt die Datei in ein Array) und `implode` (verknüpft Array-Elemente zu einer Zeichenkette). Als Verknüpfungsparemeter wird bei `implode` eine leere Zeichenkette verwendet.



Hinweis

Im Zusammenhang mit dem Umgang mit Dateien sei darauf hingewiesen, dass der Zugriff immer sequenziell erfolgt. Sie können also niemals auf direktem Weg mittendrin Zeilen einfügen oder an den Anfang setzen. »Anhängen« ist wörtlich zu nehmen, Daten können nur am Ende angefügt werden.

Ebenso problematisch ist das gezielte Löschen einer Zeile. Nur mit der Kombination mehrerer Befehle und entsprechendem Leistungsbedarf des Skripts ist dies möglich. Das folgende Beispiel zeigt die prinzipielle Vorgehensweise:

```
<?php
$datei = "muster.txt";           # Name der Datei
echo "Original:<br>";
readfile($datei);
$line = 4;                       # Zu löschende Zeile
$myfile = file($datei);          # Einlesen in ein Array
unset($myfile[$line]);           # Löschen des Elements
$fh = fopen($datei, "w");        # Überschreibend öffnen
fputs($fh, implode("", $myfile)); # Array in String
fclose($fh);
echo "<hr>Bearbeitete Datei:<br>";
readfile($datei);
?>
```

*Listing 4.33:
file_delline: Löschen
einer Zeile in einer
Datei (die Datei auf
der CD löscht nicht
das Original und
sieht deshalb etwas
anders aus)*

Das ist natürlich zugleich ein flexibler Weg, mit Dateien umzugehen, denn Sie können mit den Arrayfunktionen komfortabel die Daten manipulieren. Große Datenmengen sind aber damit trotzdem nur schwer handhabbar. Verwenden Sie dann besser Datenbanken, die auf diese Art der Manipulation spezialisiert sind.

Wenn Sie nur lesend arbeiten, können Sie auch die internen Dateizeiger nutzen. Beim Schreiben finden diese keine Berücksichtigung.

Übersicht Dateizeiger-Funktionen

Für Operationen mit Dateizeigern verwenden Sie die folgenden Funktionen:

- feof
- ftell
- fseek
- frewind

Den Dateizeiger setzen

Bei den gezeigten Beispielen wurde ausgenutzt, dass der Dateizeiger beim Lesen automatisch weiter wandert und beim Schreiben an das Ende der geschriebenen Zeichen gesetzt wird. Wenn Sie sich aber in einer Textdatei frei bewegen möchten, ist eine gezielte Positionierung des Zeigers sinnvoll.

Bei solchen Bewegungen ist es wichtig, nicht versehentlich über das Dateiende hinaus zu lesen. In diesem Fall bricht entweder die genutzte Funktion ab oder PHP reagiert mit einem Laufzeitfehler. Das Dateiende können Sie mit feof ermitteln:

```
if (feof($handle)) {
    break;
}
```

Die Funktion `feof` gibt `TRUE` zurück, wenn das Ende der Datei erreicht wurde.

**`ftell()`
`fseek()`
`frewind()`**

Die Funktion `ftell` gibt die aktuelle Position des Dateizeigers an. Sie können sich damit eine bestimmte Position merken, andere Operationen ausführen und dann die alte Position mit `fseek` wiederherstellen:

*Listing 4.34:
file_ftell: Umgang
mit dem Dateizeiger*

```
<?php
$file = 'muster.txt';
$handle = fopen($file, 'r');
rewind($handle);
fgets($handle, 20);
echo '<br>Position des Zeigers: ' . ftell($handle);
echo '<br>Ausgabe: ';
$pointer = ftell($handle);
for ($i = 0; $i <= 25; $i++) {
    echo fgetc($handle) . '&nbsp;';
}
echo '<br>Position des Zeigers: ' . ftell($handle);
fseek($handle, $pointer);
echo '<br>Position des Zeigers: ' . ftell($handle);
?>
```

Wie es funktioniert Um einen definierten Anfangszustand herzustellen, wird der Dateizeiger mit `rewind` auf den Anfang der Datei gesetzt. Zum Bewegen des Dateizeigers werden Daten aus der Datei mit `fgetc` (Zeichenweise) bzw. `fgets` (bestimmte Zeichenzahl oder bis Zeilenende) gelesen.

Dateieigenschaften ermitteln

`is_dir()`

`is_dir` ermittelt, ob der Name ein Verzeichnis ist. An der Art des Namens selbst kann man nicht feststellen, ob es eine Datei oder ein Verzeichnis ist.

`is_file()`

Die Funktion `is_file` ermittelt, ob es sich um eine reguläre Datei handelt. Dies gilt auf einem Windows-System für alle Dateien. Unix-Systeme können noch zwischen Dateien und Links unterscheiden.

`is_link()`

`is_link` ermittelt, ob es sich um einen Link (Verweis) handelt. Die unter Windows üblichen Verknüpfungen sind keine echten, sondern nur simulierte Links. PHP erkennt diese als Datei, nicht als Link.

`filemtime()`

Die Funktion `filemtime` gibt das Datum zurück, an dem die Datei das letzte Mal geändert wurde. Der Rückgabewert ist der Unix-Zeitstempel, muss also für die Ausgabe formatiert werden, beispielsweise mit `date()`.

`fileowner()`

`fileowner` holt den Namen des Besitzers der Datei. Die Funktion kann unter Windows 9x/NT nicht eingesetzt werden.

`fileperms` holt die Dateiattribute. Jedes Attribut belegt ein Bit in dem 2 Byte breiten Bitfeld. Wenn Sie kompatible Anwendungen schreiben möchten, können Sie nicht alle Unix-Attribute verwenden. Andererseits werden auch die speziellen Dateiattribute von Windows NT (beispielsweise *System* oder *Komprimiert*) nicht ermittelt. Als kleinster gemeinsamer Nenner gelten die folgenden Werte:

- 1. Die Datei ist ausführbar (Test mit `is_executable`)
- 2. Schreiben ist erlaubt (Test mit `is_writeable`)
- 4. Lesen ist erlaubt (Test mit `is_readable`)

Um das entsprechende Attribut einzeln auswerten zu können, stehen die Funktionen `is_executable`, `is_readable` und `is_writeable` zur Verfügung. Der Rückgabewert ist `TRUE`, wenn die entsprechende Eigenschaft vorhanden ist.

Wenn Sie `fileperms` anwenden möchten, maskieren Sie den Wert mit einem logischen UND. Als Maskenwert geben Sie eine Kombination aus den drei Einzelwerten an. Sind alle Rechte erforderlich, entspricht das 7 (1 + 2 + 4). Wenn Sie jedes Recht einzeln anzeigen lassen möchten, könnte die folgende Lösung interessant sein:

```
echo (fileperms($entry) & 4) ? "R" : "-";
echo (fileperms($entry) & 2) ? "W" : "-";
echo (fileperms($entry) & 1) ? "X" : "-";
```

Sie finden diese Form zusammen mit anderen Funktionen dieses Abschnitts im Projekt `TREEINFO.PHP` auf den Supportseiten zu diesem Buch und unter `ANWENDUNGEN/TREEINFO` auf der CD.



`filesize` gibt die Größe der Datei in Byte zurück. Nutzen Sie die folgende Funktion, um Umrechnungen in KByte oder MByte vorzunehmen:

fileperms()
is_executable()
is_readable()
is_writeable()

filesize()

```
<?php
function GetRealVolume($v = 0) {
    if ($v > pow(2,10)) {
        if ($v > pow(2,20)) {
            $r = (integer)($v / pow(2,20));
            $r .= " MB";
        } else {
            $r = (integer)($v / pow(2,10));
            $r .= " KB";
        } // end if
    } else {
        $r = (string) $v . " Byte";
    } // end if
    return $r;
}
$datei = "muster.txt";
```

Listing 4.35:
getvol: Umrechnung
 von Byte in KB/MB

```
echo GetRealVolume(filesize($datei));  
?>
```

filetype()

filetype gibt den Dateityp zurück. Zulässige Werte sind: fifo, char, block, link, dir und file. Der Typ link funktioniert nur bei echten Dateilinks, wie sie unter Unix verwendet werden. Die »Simulation« unter Windows für Verknüpfungen wird auch als solche erkannt und der Dateityp file ausgegeben.

Noch mehr Dateieigenschaften

In Bezug auf die Dateieigenschaften ist PHP etwas redundant. Der Vollständigkeit halber soll trotzdem auf alternative Möglichkeiten hingewiesen werden. Manchmal bieten sich so elegantere Lösungen bei der Arbeit mit Dateien.

stat()

Die Funktion stat liefert ein Array zurück, das die folgenden Werte enthält:

- 1. *device*
Gibt die Nummer des Massenspeichergeräts an. Unter Windows gibt es eine feste Zuordnung zu den Laufwerksbuchstaben: A=0, B=1, C=2 usw.
- 2. *inode*
Nummer des Inode.
- 3. *inode protection mode*
Zugriffsrechte auf Inode.
- 4. *number of links*
Anzahl der mit der Datei bestehenden Verknüpfungen.
- 5. *user id of owner*
ID des Besitzers.
- 6. *group id of owner*
ID der Gruppe des Besitzers.
- 7. *device type of inode*
Gerätetyp des Inode.
- 8. *size in byte*
Größe der Datei in Byte.
- 9. *time of last access*
Datum des letzten Zugriffs auf die Datei

- 10. *time of last modification*
Datum der letzten Änderung der Datei.
- 11. *time of last change*
Datum der letzten Dateibewegung.
- 12. *blocksize I/O*
Blockgröße des Massenspeichergeräts.
- 13. *numbers of blocks allocated*
Anzahl der von der Datei in Anspruch genommenen Blöcke.

Unter Windows können Sie nur die Eigenschaften 1, 8, 9, 10 und 11 problemlos verwenden. Beachten Sie das beim Erstellen kompatibler Anwendungen. Unter Unix stehen alle Optionen zur Verfügung.



Die Statusinformationen werden von PHP in einem internen Cache gehalten. Wenn Sie eine Datei mehrfach abfragen, wird sich die Antwort nicht ändern, auch wenn sich die Dateieigenschaften geändert haben. Sie können diesen internen Puffer mit der folgenden Funktion löschen und dann die Eigenschaften erneut abfragen:

clearstatcache()

```
clearstatcache();
```

Wenn Sie damit rechnen können, dass sich die Eigenschaften während der Ausführung eines Skripts nicht ändern, sollten Sie diesen Puffer in Anspruch nehmen. Dateizugriffe benötigen einen erheblichen Teil der Systemleistung, vor allem, wenn viele Nutzer parallel zugreifen. Denken Sie bei der Auswahl eines Lösungsweges daran, dass sich Dateien zum Speichern von Daten nur bedingt eignen. Schneller sind Datenbanken, die oft über ausgefeilte Puffermechanismen verfügen.

4.6.3 Dateienoperationen

Um mit Dateien effektiv arbeiten zu können, sind Funktionen zum Verschieben, Kopieren, Löschen und Umbenennen notwendig. Diese Funktionen werden nachfolgend vorgestellt.

Funktionsübersicht

- copy
- rename
- unlink

Anwendungsbeispiele

copy()

Die einfachen Dateifunktionen setzen voraus, dass Sie genau wissen, wie die zu behandelnden Dateien heißen. Pfadangaben können nach Bedarf eingesetzt werden. Das folgende Beispiel legt zu einer Datei eine Sicherheitskopie an und ändert dabei die Dateierweiterung.

*Listing 4.36:
file_copy: Kopieren
einer Datei mit
Umbenennung*

```
<?php
$file = "muster.txt";
echo "Kopiere Datei <b>$file</b>";
if (!@copy($file, "logdata/$file.bak")) {
    echo "<p>Datei $file nicht gefunden";
} else {
    echo "<p>Datei $file erfolgreich gesichert";
}
?>
```

Wenn Sie das Skript ausführen, benötigen Sie Schreibrechte für den Webnutzer im Verzeichnis /LOGDATA. Falls die vorangegangenen Beispiele funktionierten, sollte dies bereits eingestellt sein.

Die Funktion copy gibt True zurück, wenn der Kopiervorgang erfolgreich war. Andernfalls wird FALSE zurückgegeben. PHP gibt außerdem eine Warnung aus, die Sie unterdrücken können, indem Sie ein @ vor den Funktionsaufruf stellen:

```
if (!@copy($file, "logdata/$file.bak"))
```

Beim copy-Befehl sind Platzhalterzeichen leider nicht erlaubt. Im nächsten Abschnitt finden Sie Hinweise, wie Sie mit ganzen Verzeichnissen umgehen.

rename()

Die Funktion rename benennt Dateien um. Ein anderer Einsatz ist das Verschieben von Dateien, denn rename kann ebenso wie copy mit vollständigen und auch relativen Pfaden umgehen. Stimmt der Pfad von Quelle und Ziel nicht überein, wird die Datei verschoben, auch wenn der Dateiname gleich bleibt.

*Listing 4.37:
file_rename:
Umbenennen einer
Datei mit rename*

```
<?php
$file = "logdata/test.txt.bak";
echo "Verschiebe Datei: $file;";
$new = "logdata/test.txt";
if (!rename($file, $new)) {
    echo "<p>Datei $file nicht gefunden";
} else {
    echo "<p>Datei $file erfolgreich nach $new verschoben";
}
?>
```

Bedenken Sie auch hier, dass der Nutzer im Zielverzeichnis Schreibrechte benötigt.

Windows-Nutzern mag die Bezeichnung `unlink` wenig erklärlich sein. Die Funktion löscht Dateien. Im Handbuch gibt es wenigstens den Dummy-Eintrag *delete* dafür. Das ist insofern verwunderlich, als eine ganze Reihe anderer Funktionen tatsächlich auch doppelt vorhanden ist. `unlink` benötigt eine absolute oder relative Pfadangabe, wenn die zu löschende Datei nicht im aktuellen Verzeichnis liegt: **unlink()**

```
unlink("logdata/test.txt");
```

Auch diese Funktion gibt im Erfolgsfall `TRUE` zurück, sonst `FALSE`.

4.6.4 Umgang mit Verzeichnissen

Hier und im Folgenden wird immer von Verzeichnissen gesprochen, um über einheitliche Termini für alle Plattformen zu verfügen. Unter Windows sind damit »Ordner« (Folder) gemeint.

Funktionsübersicht

Die folgende Liste zeigt alle Funktionsnamen, die beim Umgang mit Verzeichnissen von Bedeutung sind.

- `chdir`
Wechselt das aktuelle Verzeichnis. Die Funktion verlangt eine eindeutige Pfadangabe.
- `dir`
Eine Klasse, die mit Verzeichnissen umgeht.
- `closedir`
Diese Funktion schließt ein Verzeichnis, benötigt wird ein Handle.
- `mkdir`
Erzeugt ein neues Verzeichnis und gibt ein Handle darauf zurück.
- `opendir`
Diese Funktion öffnet ein Verzeichnis und gibt ein Handle darauf zurück.
- `readdir`
Liest den nächsten Eintrag aus einem Verzeichnis anhand des übergebenen Handles.
- `rewinddir`
Setzt den Zeiger für `readdir` wieder an den Anfang.

- `rmdir`

Entfernt ein Verzeichnis.

Verarbeitung von Verzeichnissen mit dem Pseudoobjekt `dir`

`dir()`

Das Pseudoobjekt `dir` ist eine etwas genauere Betrachtung wert, denn mit den verfügbaren Methoden lassen sich etwas elegantere Lösungen schreiben als mit dem `Dateihandle`. Beachten Sie, dass dieses Objekt nicht instanziiert werden muss, sondern dies selbst bei der Zuweisung erledigt – Sie brauchen also nicht `new` anzugeben. Deshalb wurde hier auch der Begriff »Pseudoobjekt« und nicht »Klasse« verwendet.

Eine Instanz des Objekts erzeugen Sie einfach mit:

```
$folder = dir("/");
```

**`dir->handle`
`dir->path`**

Sie können nun auf zwei Eigenschaften zugreifen:

```
$folder->handle
```

Diese Eigenschaft liefert ein `Handle` zurück, mit dem Sie die übrigen Funktionen außerhalb der Klasse ansprechen können.

```
$folder->path
```

Der vollständige Pfad, der das Verzeichnis klassifiziert, wird von dieser Eigenschaft geliefert.

**`dir->read`
`dir->rewind`
`dir->close`**

Mit den Methoden `read`, `rewind` und `close` können Sie die Elemente des Verzeichnisses abrufen. An dieser Stelle werden Dateien und Unterverzeichnisse nicht unterschieden. Die nachfolgend vorgestellte Musterapplikation zeigt, wie man damit umgeht.

```
$folder->read
```

Diese Methode liest den nächsten Eintrag im Verzeichnis und gibt ihn als Zeichenkette zurück. Wenn kein Eintrag mehr existiert, wird `FALSE` zurückgegeben. Dateifunktionen, die einen Namen erwarten und kein `Handle`, können diesen Wert direkt lesen. Sie können deshalb Folgendes schreiben:

```
$entry = $folder->read;
echo $entry;
echo filesize($entry);

$folder->rewind
```

Diese Methode setzt den Zeiger wieder an den Anfang zurück. Verwechseln Sie das nicht mit `rewinddir`!

```
$folder->close
```

Schließt das Verzeichnis wieder. Das Öffnen erfolgt implizit mit dem Anlegen des Objekts. Vergessen Sie das Schließen nicht, offene Verzeichnisse verbrauchen Systemressourcen.

Im Verzeichnissystem bewegen

Um sich im Verzeichnissystem bewegen zu können, reichen die Verzeichnisfunktionen aus. Die Anwendung ist dennoch nicht trivial, denn es sind viele Sonderfälle zu betrachten, wenn man sich im Verzeichnisbaum frei bewegt. Das folgende Beispiel startet die Anzeige in der Root und bewegt sich auf dem aktuellen Laufwerk frei in allen Verzeichnissen, die entsprechende Zugriffsrechte haben. Bei Dateien werden Namen, Größe in Byte und das Datum angezeigt.

```
<?php
function GetRealVolume($v = 0) {

    # Definition aus Listing 4.35 auf Seite 363

}
$strSelFolder = stripslashes($folder);
if ($strSelFolder == '') {
    $strSelFolder = "/";    // -> Rootvolume
}
echo <<<HEADER
    <style>
        .treetext {font-size:10pt}
    </style>
    <p class=treetext>Aktueller Pfad: <b>$strSelFolder</b>
    <hr noshade width=400 size=1 align=left>
HEADER;
@chdir($strSelFolder);
$cdir = @dir($strSelFolder);

if (!is_object($cdir)) {
    die("<span style=color:red class=treetext>Keine Zugriffsrechte
        in diesem Verzeichnis</span></div></body></html>");
}

echo <<<HEAD1
    <table border=1 bordercolordark=#eeeeee
        bordercolorlight=#eeeeee bordercolor=#eeeeee
        cellpadding=1 cellspacing=0>
HEAD1;

while ($entry = $cdir->read()) {
    if (!preg_match("/^\.$/", $entry)) {
        if (is_dir($entry)) {
            if (preg_match("/^\.$/", $entry)) {
                $folder = urlencode(substr($strSelFolder, 0,
                    strrpos($strSelFolder, "/")));
```

Listing 4.38:
tree: Ein Skript, mit dem Sie sich per Browser frei zwischen Verzeichnissen bewegen können.

```

    } else {
        $folder = urlencode($strSelFolder);
        strlen($strSelFolder)==1 ? $sep= "" : $sep= "/";
        $folder .= $sep;
        $folder .= urlencode($entry);
    }
    $type = strtoupper filetype($entry);
    echo <<<TREEROW
        <tr><td nowrap class=treetext>
            <a href="tree?folder=$folder">$entry</a>
        </td><td class=treetext>$type</td></tr>
TREEROW;
    }
}
}
echo "</table>";

echo <<<HEAD2
    <hr noshade width=400 size=1 align=left>
    <span class=treetext>
        Dateien in diesem Verzeichnis und ausf&uuml;rliche
        Informationen &uuml;ber diese Dateien:
    </span>
    <hr noshade width=400 size=1 align=left>
HEAD2;
echo <<<HEAD3
    <table border=1 bordercolordark=#eeeeee
        bordercolorlight=#eeeeee bordercolor=#eeeeee
        cellpadding=1 cellspacing=0>
    <tr bordercolor=Black class="treetext">
        <th>Name</th><th>Gr&ouml;&szlig;</th>
        <th>Datum</th><th>Typ</th>
    </tr>
HEAD3;
$cdire->rewind();
clearstatcache();
$i = 0;
while ($entry = $cdire->read()) {
    if (!is_dir($entry)) {
        if ($i++ % 2 == 0) {
            echo "<tr bgcolor=LightGrey>";
        } else {
            echo "<tr bgcolor=white>";
        }
    }
    echo "<td valign=top class=treetext>$entry</td>";
    echo "<td nowrap class=treetext align=right>" .
        GetRealVolume(filesize($entry))."</td>";
    echo "<td nowrap class=treetext>" .
        date("j.M Y H:i:s",filetime($entry))."</td>";
    echo "<td class=treetext>";
    if (is_link($entry)) { echo "[LNK]"; }
}

```

```
        if (is_file($entry)) { echo "[File]"; }
        echo '</td></tr>';
    }
}
echo '</table>';
$cdire->close;
```

Zuerst werden die automatisch eingefügten Backslashes entfernt. Dieser Effekt tritt nur auf der Windows-Plattform auf, wo die Verzeichnistrenner als Backslash ausgeführt werden.

Wie es funktioniert

```
$strSelFolder = stripslashes($folder);
```

Außerdem wird als Startverzeichnis ROOT ausgewählt:

```
if ($strSelFolder == '') {
    $strSelFolder = "/";
}
```

Um plattformunabhängig programmieren zu können, werden neue Verzeichnisse immer nur mit dem normalen Trennzeichen »/« gebildet. Solchermaßen zum Glück gezwungen, kommt auch Windows damit zurecht, bei Unix ist es ohnehin Standard.

Dann wird das aktuelle Verzeichnis ausgewählt (chdir) und ein Objekt darauf gebildet (dir):

```
chdir($strSelFolder);
$cdire = dir($strSelFolder);
```

Im nächsten Schritt werden alle Einträge in diesem Verzeichnis durchlaufen. Jeder Eintrag wird mit der read-Methode des dir-Objekts geholt. Die Methode gibt FALSE zurück, wenn es nichts mehr zu lesen gibt. Zuweisung der Einträge und Abfrage der Schleifenbedingung können deshalb zusammengefasst werden:

```
while ($entry = $cdire->read()) {
```

Leider sind die Einträge vollkommen unsortiert, Verzeichnisse und Dateien stehen bunt durcheinander. Mit der Funktion is_dir wird erkannt, ob es sich um ein Verzeichnis handelt:

```
if (is_dir($entry)) {
```

Hier erfolgt die Anzeige der Verzeichnisse. Die zweite Schleife macht es genau umgekehrt, hier werden nur Dateien angezeigt. Dazwischen muss der interne Zeiger folgendermaßen zurückgesetzt werden:

```
$cdire->rewind();
```

Knifflig ist die Berechnung der Links; die tiefer- und höhergehenden Verzeichnisse müssen erreicht werden. Eine Ebene höher geht es mit:

```
substr($strSelFolder,0,strrpos($strSelFolder,"/"));
```

Intern wird das übergeordnete Verzeichnis durch »..<« dargestellt, die Erkennung erfolgt mit der folgenden Zeile:

```
if (preg_match("/^\.\.$/", $entry)) {
```

Die urlencode-Funktion ist notwendig, um auch Verzeichnisnamen verarbeiten zu können, die Leerzeichen enthalten:

```
$folder = urlencode(substr($strSelfFolder, 0 ,  
                        strrpos($strSelfFolder, "/")));
```

4.6.5 Dateiupload

Zu den spannenden Funktionen gehört auf jeden Fall das Dateiupload per Browser. Es spart bei wenigen Zugriffen einen zusätzlichen FTP-Client. Für Nutzer besteht eine einfache Möglichkeit, Dateien abzulegen, ohne dass der Server selbst geöffnet werden muss und dadurch möglicherweise Sicherheitslöcher entstehen. Neben der Unterstützung im Browser werden auch Dateifunktionen verwendet.

Theoretischer Hintergrund

Das Upload-Verhalten selbst ist nicht standardmäßig in HTTP realisiert. Erst neuere Browser beherrschen die entsprechenden Tags, mit denen die Auswahl der Dateien von der lokalen Festplatte erfolgt.

RFC 1867

Die Definition des Dateiupload per HTTP erfolgte in der RFC 1867. Dabei wird die Methode POST so erweitert, dass eine Ziel-URL übertragen werden kann. Die Daten selbst werden mit dem Dateityp »multi-part/form-data« gesendet. Das Ablegen der Dateien erfolgt also unter einer URL.

Das Versenden von Dateien erfolgt also im Wesentlichen über ein Formular. Der Aufbau kann sich an folgendem Muster orientieren:

*Listing 4.39:
Formular zum
Hochladen von
Dateien*

```
<form enctype="multipart/form-data" action="_URL_" method=post>  
<input type="hidden" name="MAX_FILE_SIZE" value="1000">  
Ihre Dateiauswahl: <input name="userfile" type="file">  
<input type="submit" value="Datei senden">  
</form>
```

Wie es funktioniert

Drei Unterschiede zu den bereits beschriebenen Formularen sind hier zu beachten. Zum einen muss der Dateityp angegeben werden. Hierzu wird in HTML das Attribut ENCTYPE genutzt:

```
<form enctype="multipart/form-data" action="_URL_" method=post>
```

Zusätzliche Optionen können in versteckten Feldern aktiviert werden, beispielsweise die Begrenzung der maximalen Dateigröße:

```
<input type="hidden" name="MAX_FILE_SIZE" value="1000">
```

Die Auswahl der Datei ist Sache des Browsers, er muss den Formular-typ FILE unterstützen. Diese Eingabefelder sehen aus wie Textfelder und bieten zusätzlich eine Schaltfläche, die mit BROWSE... oder DATEI... beschriftet ist:

```
<input name="userfile" TYPE="file">
```

Als Ziel-URL geben Sie die URL eines PHP-Skripts an, der Name `_URL_` im Beispiel ist nur ein Platzhalter. In diesem Skript werden automatisch mehrere Variablen definiert, die Werte über den Upload-Vorgang enthalten:

- `$userfile`

Der Name der Datei im temporären Verzeichnis.

- `$userfile_name`

Der Name der Originaldatei.

- `$userfile_size`

Die tatsächliche Größe der Datei.

- `$userfile_type`

Der Dateityp im MIME-Format, also beispielsweise »image/gif« für GIF-Bilder.

Der konkrete Name der Variablen kann mit dem Namen des Dateifeldes beeinflusst werden. Wenn Sie folgendes Tag schreiben, würden die Variablen `$upload_name` usw. heißen:

```
<input name="upload" TYPE="file">
```

Die hochgeladenen Dateien werden in einem temporären Verzeichnis des Servers gespeichert. Am Ende des Skripts wird dieses Verzeichnis gelöscht, Sie müssen die hochgeladenen Dateien also sofort an ihren Bestimmungsort kopieren. Dies zu vergessen ist einer der häufigsten Fehler. Der Sinn ist die mögliche Vorbehandlung der Dateien vor der Ablage im Zielverzeichnis. So könnten Sie die Größe auswerten und zu große Dateien ablehnen.

Universeller Zugriff auf die Dateiinformationen

Neben der direkten Übergabe der Informationen in Variablen steht auch ein Array mit diesen Informationen zur Verfügung. Sie können dieses Array verwenden, wenn die Dateinamen nicht bekannt sind oder sehr viele Dateien hochgeladen werden. Das folgende Beispiel, das in modifizierter Form im Projekt *phpTemple* zu finden ist, zeigt die Anwendung.

**\$HTTP
_POST_FILES**

Listing 4.40:
uniupload:
 Universelles Upload-
 Skript

```

Wie viele Dateien möchten Sie hochladen?
<form action="<?php echo $PHP_SELF; ?>" method="post">
<select name="numfiles" size="1">
  <option value="1">1 Datei
  <option value="2">2 Dateien
  <option value="3">3 Dateien
  <option value="5">5 Dateien
  <option value="10">10 Dateien
</select>
<input type="Submit" value="Anzahl festlegen" name="regnum">
</form>
<form enctype="multipart/form-data" method="post" action="<?php
echo $PHP_SELF; ?>">
<?php
if (isset($regnum)) {
  echo "<b>$numfiles</b> ";
  echo $numfiles == '1' ? 'Datei kann' : 'Dateien können';
  echo ' hochgeladen werden:<p>';
  for ($i=1; $i<=$numfiles; $i++) {
    echo "<input type='File' name='myfile$i'><br>\n";
  }
  echo '<p>';
  echo "<input type='Submit' name='sendfiles'
    value='Dateien senden'>";
}
if (isset($sendfiles)) {
  $numsendfiles = count($_HTTP_POST_FILES);
  echo "<b>$numsendfiles</b> ";
  echo $numsendfiles == 1 ? 'Datei' : 'Dateien';
  echo ' wurde gesendet.<p>';
  foreach($_HTTP_POST_FILES as $strFieldName => $arrPostFiles) {
    if ($arrPostFiles['size'] > 0) {
      $strFileName = $arrPostFiles['name'];
      $intFileSize = $arrPostFiles['size'];
      $strFileMIME = $arrPostFiles['type'];
      $strFileTemp = $arrPostFiles['tmp_name'];
      @copy ($strFileTemp, "upload/$strFileName");
      echo "Datei <b>$strFileName</b>
        erfolgreich hochgeladen:";
      echo "<ul>";
      echo "<li>Größe: $intFileSize Bytes<br>";
      echo "<li>MIME: $strFileMIME<br>";
      echo "</ul>";
    } /* end if */
  } /* end foreach */
}
?>
</form>

```

Wie es funktioniert

Zuerst wird nur ein Formular angezeigt, mit dem der Nutzer die Anzahl der Dateien bestimmen kann, die er hochladen will:

```
<form action="<?php echo $PHP_SELF; ?>" method="post">
<select name="numfiles" size="1">
  <option value="1">1 Datei
  ...
  <option value="10">10 Dateien
</select>
```

Universeller Upload

Wie viele Dateien möchten Sie hochladen?

1 Datei

Abbildung 4.5:
Auswahl der Anzahl
der Dateien

Wenn dieses Formular abgesendet wurde, wird die Feldliste erstellt:

```
for ($i=1; $i<=$numfiles; $i++) {
  echo "<input type='File' name='myfile$i'><br>\n";
}
```

Universeller Upload

Wie viele Dateien möchten Sie hochladen?

1 Datei

2 Dateien können hochgeladen werden:

Abbildung 4.6:
Automatisch erstellte
Feldliste

Die Informationen, die mit Hilfe des zweiten Formulars gesendet wurden, werden dann dem Array `$HTTP_POST_FILES` entnommen:

```
foreach($HTTP_POST_FILES as $strFieldName => $arrPostFiles) {
  if ($arrPostFiles['size'] > 0) {
    $strFileName = $arrPostFiles['name'];
    $intFileSize = $arrPostFiles['size'];
    $strFileMIME = $arrPostFiles['type'];
    $strFileTemp = $arrPostFiles['tmp_name'];
```

Die erkannten Dateien werden dann in das Zielverzeichnis (hier ein Unterverzeichnis `/UPLOAD`) kopiert.

```
copy ($strFileTemp, "upload/$strFileName");
```

Der Rest des Skriptes dient lediglich der Benutzerführung. Die folgende Abbildung zeigt, wie zusätzliche Informationen über Größe und Art der Datei angezeigt werden. Sie können diese Daten auch verwenden, um das Hochladen zu steuern.

Abbildung 4.7:
Erfolgreich
hochgeladene Dateien

Universeller Upload
 Wie viele Dateien möchten Sie hochladen?

1 Datei

Anzahl festlegen

 2 Dateien wurde gesendet.
 Datei **247Mail_Liste-Sunbelt.pdf** erfolgreich hochgeladen:

- Größe: 28977 Bytes
- MIME: application/pdf

 Datei **Fernsehen.doc** erfolgreich hochgeladen:

- Größe: 35328 Bytes
- MIME: application/msword

Wie das Hochladen funktioniert

Das Hochladen wird von PHP ausgeführt. Empfangene Daten werden in einem temporären Verzeichnis gespeichert und müssen unmittelbar danach an den Bestimmungsort kopiert werden. Am Ende eines Skripts wird das temporäre Verzeichnis geleert – Sie können in folgenden Skripten nicht mehr auf die Dateien zugreifen. Für das Kopieren gibt es zwei Funktionen:

- copy

Diese Funktion ist bei allen Versionen einsetzbar. Damit wird ein normaler Kopiervorgang ausgeführt. Sie müssen dazu den vollständigen Quellpfad angeben.

- move_uploaded_file

Diese Funktion ist erst ab PHP 4.0.2 verfügbar und vereinfacht den Transport der Daten.



Hinweis

Das temporäre Verzeichnis, wo die Dateien kurzzeitig abgelegt werden, muss aber in jedem Fall existieren. Die Standardeinstellung in der Datei PHP.INI ist /TMP. Wenn Sie unter Windows arbeiten, legen Sie entweder im Stammlaufwerk ein Verzeichnis \TMP an oder ändern die Einstellung in der Konfiguration auf C:\TEMP.

Größe begrenzen

Sie können die Größe der übertragenen Dateien auch auf direktem Wege begrenzen. Zum einen eignet sich dazu das versteckte Feld mit dem Namen MAX_FILE_SIZE. Der Wert des Feldes bestimmt die Größe in Byte. Größere Dateien werden nach dem Upload abgelehnt. Der Uploadvorgang selbst lässt sich nicht unterbinden, da der Browser die Formulardaten und die Datei in einer Anforderung sendet.

php.ini

Es gibt zusätzlich in der Konfigurationsdatei PHP.INI den folgenden Eintrag:

```
upload_max_filesize = 2000000;
```

Dies ist der Standardwert für die Begrenzung der Dateigröße: 2 MByte. Der mit `MAX_FILE_SIZE` eingestellte Wert kann nie größer sein als der in den Grundeinstellungen eingetragene. Wenn Sie PHP als Modul im Apache nutzen, können Sie auch dort den Wert mit Hilfe der Direktive `php_upload_max_filesize` einstellen.

Zugriffsrechte auf das Dateisystem

Generell müssen Sie bedenken, dass beim Kopieren der Dateien vom temporären Verzeichnis in das Zielverzeichnis entsprechende Rechte notwendig sind. Der Nutzer, der Uploads auslöst, muss Dateien erzeugen und schreiben dürfen. Wenn Sie den anonymen Zugriff auf die Upload-Funktion erlauben, sind entsprechende Vorkehrungen nötig, um keine Sicherheitslücken entstehen zu lassen. Bedenken Sie beispielsweise, dass Nutzer ausführbare Dateien, trojanische Pferde oder Viren hochladen könnten und dann zur Ausführung bringen. ➡ Abschnitt 4.8 *Sicherheit* ab Seite 400 geht auf die Sicherheit und das Rechtesystem unter Unix und Windows ein.

Mehrere Dateien hochladen

Oft ist es komfortabler, in einem Zug gleich mehrere Dateien zu übertragen. An der Funktionsweise ändert sich dabei nichts. Sie deklarieren die Formularfelder einfach als Arrays. Jede Datei belegt dann ein Element des Arrays. Das bereits gezeigte Beispiel würde dann folgendermaßen aussehen:

```
<FORM ENCTYPE="multipart/form-data" ACTION=" _URL_ " METHOD=POST>
<INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="1000">
Ihre 1. Datei: <INPUT NAME="userfile[]" TYPE="file">
Ihre 2. Datei: <INPUT NAME="userfile[]" TYPE="file">
Ihre 3. Datei: <INPUT NAME="userfile[]" TYPE="file">
<INPUT TYPE="submit" VALUE="Datei senden">
</FORM>
```

Listing 4.41:
Hochladen mehrere
Dateien

Die Abfrage erfolgt dann über die Variablen `$userfile_name[0]` für die erste Datei, `$userfile_name[1]` für die zweite usw.

Upload mit der PUT-Methode

Eine eher selten angewandte Methode ist PUT. Bislang war immer nur von GET und POST die Rede. Auch PUT ist eine HTTP-Methode und dient eigentlich genau diesem Zweck. Allerdings müssen auch die Browser eine entsprechende Unterstützung bieten. Leider ist diese kaum gegeben, entsprechend selten die Anwendungen der PUT-Methode. Für den Datei-Upload können Sie derzeit den Netscape-Composer einsetzen, ein einfaches Gestaltungswerkzeug mit

WYSIWYG¹⁹-Darstellung. Daneben unterstützt der W3C-Browser Amaya diese Methode noch. Dies ist umso verwunderlicher, als die Anwendung einfacher ist als mit POST und diversen Erweiterungen. Die HTTP-Methode wird etwa folgendermaßen angesprochen:

```
PUT /pfad/dateiname.html HTTP/1.1
```

Dieser Befehl platziert die Datei im angegebenen Pfad. Aus Sicherheitsgründen sollte auch hier »wildes Upload« unterbunden werden. Im Apache können Sie die folgende Serverdirektive in die Konfigurationsdatei aufnehmen:

```
Script PUT /put.php
```

Damit werden alle Dateien zuerst an das so benannte Skript gesendet. Sie können dann über bestimmte Variablen auf die Daten der Datei zugreifen und entscheiden, ob ein Upload zulässig ist oder nicht. Ein Beispiel wäre die folgende Zeile zum Kopieren der Datei:

```
<?php
copy ($PHP_UPLOADED_FILE_NAME, $DOCUMENT_ROOT.$REQUEST.URI);
?>
```

Listing 4.42:
Hochladen mit PUT
(ungetestet)

Damit wird die Datei zu der vom Client gewünschten Stelle kopiert. Durch den Zugriff auf diese Variablen vor dem Kopieren ist eine Kontrolle leicht möglich. In diesem Punkt unterscheidet sich PUT übrigens nicht mehr von POST. Auch hier gehen die Dateien am Ende des Skripts verloren, wenn sie nicht aus dem temporären Verzeichnis heraus kopiert werden.

4.7 Verbindungen zu Servern im Internet

Daten können nicht nur von der lokalen Festplatte des Webserver gelesen werden. Dieselben Funktionen eignen sich auch für den Zugriff auf entfernte Server per HTTP und FTP. Man spricht von einem sogenannten »Wrapper«, einer Umleitung des Funktionsaufrufes an eine andere Instanz zur Ausführung.

CURL

Direkter arbeiten die Funktionen der CURL-Bibliothek. CURL steht hier für *Client URL*. Diese Funktionen sind explizit für den Zugriff auf andere Server entworfen worden und bieten mehr Möglichkeiten als die Dateifunktionen. Für den Zugriff auf FTP-Server werden FTP-Funktionen verwendet.

Hinweise auf weitere Abschnitte

Die Wrapperfunktionen werden nachfolgend erklärt. Der universelle Zugriff mit darauf spezialisierten Funktionen wird in ➡ Abschnitt 4.7.3 ab Seite 386 gezeigt. Eine Einführung in die nativen FTP-

¹⁹ WYSIWYG = What You See Is What You Get, Editor mit grafischer Ansicht.

Funktionen finden Sie in ➡ Abschnitt 4.7.4 *FTP-Funktionen* ab Seite 393.

4.7.1 Die Umgebungsvariablen des Webservers

Jeder Webserver liefert sogenannte Umgebungsvariablen, die in PHP problemlos abgerufen werden können. Die folgende Liste zeigt die wichtigsten: **Informationen des Webservers**

- **SERVER_NAME**
Der Name des Hosts, beispielsweise *www.comzept.de*.
- **GATEWAY_INTERFACE**
Bezeichnung der Schnittstelle, beispielsweise *CGI/1.1*. Dieser Wert ist nur bei CGI verfügbar, nicht jedoch, wenn PHP als Apache-Modul läuft
- **SERVER_PROTOCOL**
Das Protokoll des Servers, meist *HTTP/1.1*.
- **SERVER_PORT**
Verwendeter Port, im WWW meist 80 (dies ist der Standardport für HTTP).
- **REQUEST_METHOD**
Kann GET, POST oder PUT sein, also die zuletzt verwendete Anforderung des Browsers.
- **PATH_INFO**
Die Pfadinformation des aufgerufenen Skripts. Zeigt den kompletten virtuellen Pfad an.
- **PATH_TRANSLATED**
Physischer Pfad zum Skript, wenn der Server diese Information bereit stellt.
- **SCRIPT_NAME**
Pfad und Name zum Skript, beispielsweise */cgi-bin/show-vars.php*.
- **QUERY_STRING**
Enthält die GET-Daten, beispielsweise *var=value&session=2345*, wenn die letzte Anforderung GET-Daten enthielt.
- **REMOTE_HOST**
Der Name des Clients, beispielsweise *a162.dialup.mevacom.de*.

- REMOTE_ADDR
Die IP-Adresse des Clients, beispielsweise 62.208.0.162.
- AUTH_TYPE
Authentifizierungstyp bei gesicherten Sites.
- REMOTE_USER
Anmeldename bei gesicherten Sites.
- REMOTE_IDENT
Identifikator.
- CONTENT_TYPE
Angabe des Inhaltstyps im MIME-Format, beispielsweise *text/html*.
- CONTENT_LENGTH
Länge des Inhalts in Byte.
- SERVER_SOFTWARE
So meldet sich der Server, beispielsweise *Apache/1.3.6 (Unix) PHP/3.0.11 ApacheJServ/1.0b4*.
- HTTP_ACCEPT
Enthält, was der Browser an MIME-Typen akzeptieren kann und will.
- HTTP_USER_AGENT
Kennung des Browsers, beispielsweise *Mozilla/4.0 (compatible; MSIE 5.0; Windows NT)*. Der Wert kann auf vielen Systemen manipuliert werden und ist deshalb mit Vorsicht zu betrachten.
- HTTP_COOKIE
Die Cookie-Daten, wenn Cookies gesendet wurden.
- HTTP_REFERER
Die letzte Adresse, von welcher der Browser kam. Wurde die Seite direkt aufgerufen, ist der Wert leer.

Der Abruf erfolgt in Form normaler Variablen, also `SERVER_SOFTWARE` als `$SERVER_SOFTWARE` oder durch Zugriff auf das entsprechende Array `$HTTP_SERVER_VARS`.

Listing 4.43:
servervariables:
Anzeige von
Servervariablen

```
<?php
echo '<ul>';
foreach ($HTTP_SERVER_VARS as $key => $value) {
    echo "<li><b>$key</b>: $value</li>";
}
```

```
}  
echo '</ul>';  
?>
```

Die konkrete Anzahl und Verfügbarkeit der Variablen hängt vom verwendeten Webserver und dessen Konfiguration ab. Wenn Sie Skripte portieren, verlassen Sie sich nicht darauf, dass alle Variablen zur Verfügung stehen und auch eine identische Bedeutung haben.



In der Praxis können Sie vor allem folgende Informationen verwenden:

Praxistipps

- IP-Adresse des Clients: `REMOTE_ADDR`
- Typ des Browsers: `HTTP_USER_AGENT`
- Name des eigenen Server: `SERVER_NAME` oder `HTTP_HOST`
- Seite, die vor diesem Aufruf verlassen wurde: `HTTP_REFERER`
- Request-Methode (GET oder POST): `REQUEST_METHOD`

Diese Informationen sollten auch unter allen Umständen verfügbar sein – unabhängig vom Betriebssystem und vom Webserver.

```
• ALLUSERSPROFILE: C:\Dokumente und Einstellungen\All Users  
• CommonProgramFiles: C:\Programme\Gemeinsame Dateien  
• COMPUTERNAME: SRV1  
• ComSpec: C:\WINNT\system32\cmd.exe  
• CONTENT_LENGTH: 0  
• GATEWAY_INTERFACE: CGI/1.1  
• HTTP_ACCEPT: */*  
• HTTP_ACCEPT_LANGUAGE: de  
• HTTP_CONNECTION: Keep-Alive  
• HTTP_HOST: srv1  
• HTTP_REFERER: http://srv1/book.sites/phpbooknew2e/show.php4  
• HTTP_USER_AGENT: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)  
• HTTP_COOKIE: sessionid=5f44a4622e7832139ca20e9dce6f50b; PHPSESSID=75f3269352134fce950b67f08d68c29a;  
  Answender=7f008980df8a0d4f6d13c097e5f8f9; Anwendung_3=9785f41536abc3783fcb89622ee5c5c  
• HTTP_ACCEPT_ENCODING: gzip, deflate  
• HTTPS: off  
• INSTANCE_ID: 1  
• LOCAL_ADDR: 192.168.100.12  
• NUMBER_OF_PROCESSORS: 1  
• OsLibPath: C:\WINNT\system32\os2\dll;  
• OS: Windows_NT  
• Path: C:\WINNT\system32;C:\WINNT;C:\WINNT\System32\Wbem;C:\Programme\Resource Kit\;  
• PATH_INFO: /book.sites/phpbooknew2e/process_scripts.php4  
• PATH_TRANSLATED: c:\inetpub\wwwroot\book.sites\phpbooknew2e\process_scripts.php4  
• PATHTEXT: .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.VBS  
• PROCESSOR_ARCHITECTURE: x86  
• PROCESSOR_IDENTIFIER: x86 Family 6 Model 8 Stepping 1, GenuineIntel  
• PROCESSOR_LEVEL: 6  
• PROCESSOR_REVISION: 0801  
• ProgramFiles: C:\Programme  
• QUERY_STRING: script=servervariables  
• REMOTE_ADDR: 192.168.100.25  
• REMOTE_HOST: 192.168.100.25  
• REQUEST_METHOD: GET  
• SCRIPT_NAME: /book.sites/phpbooknew2e/process_scripts.php4  
• SERVER_NAME: srv1  
• SERVER_PORT: 80  
• SERVER_PORT_SECURE: 0  
• SERVER_PROTOCOL: HTTP/1.1
```

Abbildung 4.8:
Ausgabe von
Servervariablen

4.7.2 HTTP- und FTP-Verbindungen

Um Daten per HTTP oder FTP zu laden, sind keine besonderen Befehle nötig. Die Funktion `fopen` kann als Dateiname auch ein vollständig qualifizierten URL verarbeiten. Die betreffende Site wird geöffnet und komplett gelesen.

Einsatzmöglichkeiten

Der Einsatz ist äußerst vielfältig. Sie können so auf sehr einfache Art und Weise Robots bauen, die fremde Seite durchsuchen und auswerten. Die Beispielskripte geben die nötigen Hinweise dazu.

Das folgende Beispiel zeigt eine einfache Anwendung, das eine (hier noch fest kodierte) Seite abruft und wieder ausgibt. Damit Ihr Browser nicht an den »fremden« HTML-Tags scheitert, werden spitze Klammern, Leerzeichen und Zeilenumbrüche ersetzt. Wenn Sie das Skript lokal laufen lassen, benötigen Sie eine stehende Internetverbindung. Auf einem regulären Webserver ist die ohnehin gegeben.

*Listing 4.44:
file http: Einlesen
einer Website von
einem fremden Server
und Ausgabe des
Inhalts im Browser*

```
<?php
$url = "http://www.asp.comzept.de/start.htm";
echo "Lese $url ein.<br>";
$fp = fopen($url, "r");
echo 'Zeige Ausgabe an.<p /><pre>';
while ($buffer = fgets($fp, 1000)) {
    echo htmlspecialchars($buffer);
}
echo '</pre>';
fclose($fp);
?>
```

Die Funktionen, die hier verwendet werden, sind nicht speziell für diesen Zweck gedacht. Das schon bekannte `fopen` kann neben lokalen Pfaden auch Anforderungen per *http* und *ftp* umsetzen. Bei der Bildung der Adresse müssen Sie die Syntax beachten und in jedem Fall das Protokoll explizit angeben.

FTP-Adresse

Bei FTP-Servern gilt folgender Aufbau des URL:

```
ftp://server.domain.tld/pfad/datei.ext
```

HTTP-Adresse

HTTP-Server hingegen werden folgendermaßen angesprochen:

```
http://server.domain.tld:port/pfad/datei.ext
```

Andere Protokolle können Sie hier nicht verwenden. Die Portangabe kann entfallen, wenn es sich um die Standardports handelt. Die Grundlagen zu Ports wurden bereits in Kapitel 1 behandelt. Für HTTP ist der Standardport 80, für FTP ist es 21.

Wenn die Seiten geschützt sind, können die Anmeldeinformationen direkt in der URL mit übergeben werden. Die meisten Server können das auswerten und erlauben so einem Skript den Zugriff, das ja die Dialogfelder eines Browsers nicht erzeugen kann. Schreiben Sie für FTP folgendes:

```
ftp://username:kennwort@server.domain.tld/pfad/datei.ext
```

Ebenso können Sie mit HTTP umgehen:

```
http://username:kennwort@server.domain.tld:port/pfad/datei.ext
```

Beachten Sie, dass sich durch die Übermittlung von Benutzername und Kennwort Sicherheitsrisiken ergeben können. Der URL wird im Klartext übertragen, in Proxyservern und im Cache des Webserver zwischengespeichert und steht natürlich auch im Skript, dass diese Abfrage erzeugt. Für besonders kritische Daten ist diese Technik deshalb nicht empfehlenswert.



Zugriff auf andere Ressourcen mit Socketfunktionen

PHP ist gut geeignet, auf alle Ressourcen eines Servers zugreifen zu können. Sicher kennen Sie die vielen Free-E-Mail-Angebote im Internet, die oft Browser-basierte Schnittstellen bereitstellen. Der Vorteil liegt auf der Hand. Sie können Ihre E-Mail von jedem Punkt aus nur mit einem Browser lesen. Damit sind Sie unabhängig von der Bereitstellung eines entsprechenden E-Mail-Clients.

Die üblichen Protokolle für E-Mail sind POP3 und IMAP4. Am weitesten verbreitet ist POP3. Entsprechend groß ist auch das Angebot an POP3-Servern. Die per SMTP empfangene E-Mail wird dann den Clients per POP3 zum Download zur Verfügung gestellt. Die Umsetzung auf den Browser geschieht sinnvoll im Server.

POP3 abfragen

PHP muss also in der Lage sein, einen POP3-Server abzufragen und die Daten in geeigneter Form weiterzugeben.

Der POP3-Server lauscht normalerweise auf Port 110 nach den für ihn bestimmten Kommandos. Die folgende Variante zeigt den Ablauf der Anmeldung und Authentifizierung an einem POP3-Server. Ersetzen Sie die entsprechenden Werte durch Ihren POP3-Server, sonst wird die Anmeldung zurückgewiesen. Das Skript verzichtet auf das Herunterladen und Löschen der Mail, kann also auch gefahrlos auf einem »Live«-System arbeiten. Abgesehen davon ist die Version »quick-and-dirty« programmiert, bedarf also sicher der Ergänzung. Im Praxisteil können Sie sich eine etwas ausgereifere Lösung ansehen.

fsocketopen()

```
<html>
<body>
<h1>Server abfragen</h1>
<h3>Abfragen eines POP3-Servers</h3>
<p>
<?
function command($strCommand) {
    echo $strCommand."<br>";
    return $strCommand;
}
/*
Diese drei Werte muessen auf einen POP3-Server
```

*Listing 4.45:
socket_pop3: Abfrage
eines beliebigen
POP3-Servers im
Internet*


```

eingestellt werden
*/

$server = "pop.kundenserver.de";
$user   = "m6241390-open";
$pass   = "forall";

$pop3 = fsockopen("$server","110");
if ($pop3 <= 0) echo "Fehler...";
echo fgets($pop3, 1024)."<br>";
fputs($pop3, command("USER $user\r\n"));
echo fgets($pop3, 1024)."<br>";
fputs($pop3, command("PASS $pass\r\n"));
echo fgets($pop3, 1024)."<br>";
fputs($pop3, command("STAT\r\n"));
echo fgets($pop3, 1024)."<br>";
fputs($pop3, command("QUIT\r\n"));
echo fgets($pop3, 1024)."<br>";
fclose($pop3);
?>
</body>
</html>

```

Abbildung 4.9:
Eine »öffentliche«
POP3-Abfrage



Wenn Sie das Beispiel in Listing 4.45 ausführen, wird der Ablauf des Protokolls im Browser mitgeschrieben. Der Befehl STAT zeigt die Anzahl der E-Mails und deren gesamte Größe an. Abbildung 4.9 zeigt das Ergebnis. Hier liegt eine E-Mail mit 943 Byte vor. Mit den Kom-

mandos RETR kann nun abgeholt werden, DELE löscht die E-Mails vom Server.

Das Beispiel ist zwar primitiv, zeigt aber alle nötigen Techniken. Zuerst müssen Sie sich natürlich über die Art und Weise der Kommunikation mit dem POP3-Server (oder allen anderen Servern) im Klaren sein. Die Grundlagen dazu finden Sie in ➡ Abschnitt *POP3* auf Seite 87. Die Kommunikation findet über den Austausch einfachster Kommandos statt, die Übertragung erfolgt offen im ASCII-Format.

Wie es funktioniert

Die Verbindung wird mit der Bildung des Handles vorbereitet:

```
$pop3 = fsockopen("$server", "110");
```

Wird eine Zahl kleiner als 0 zurückgegeben, ist ein Fehler aufgetreten. War die Verbindung erfolgreich, kann jetzt mit normalen Dateifunktionen auf diese Verbindung zugegriffen werden. Mit der folgenden Zeile senden Sie Text an den Server:

```
fputs($pop3, command("USER $user\r\n"));
```

Sinnvollerweise sind dies die vom Protokoll erwarteten Kommandos oder Daten. Im gleichen Puffer stellt der Server auch seine Antwort bereit. Die Abfrage erfolgt mit dieser Anweisung:

```
fgets($pop3, 1024);
```

Statt der Ausgabe im Browser mit echo wird die Zuweisung zu einer Variablen und weitere Auswertung der häufigere Weg sein. Die eigentliche Schwierigkeit besteht in der Programmierung angemessener Reaktionen auf alle erdenklichen Einsatzfälle.

Erweiterte Funktionen beim direkten Zugriff

In Abhängigkeit von der Reaktion des Servers müssen Sie über einige ergänzende Funktionen verfügen. So gibt es die Möglichkeit, dass der Server nicht sofort reagiert. Fragen Sie einen Port im Polling ab, beispielsweise mit einer Schleife, kann die Belastung für Server und Verbindung sehr groß sein. Andererseits können Sie das Skript unter Umständen nicht fortsetzen, wenn keine Antwort vorliegt.



Die Funktion `set_socket_blocking` kann zur Kontrolle des Verhaltens eingesetzt werden. Im POP3-Beispiel war dies nicht notwendig, der Server wird jede Anfrage beantworten, notfalls mit Fehlermeldung. Fällt der Server aber aus, kann sich das Skript aufhängen, zumindest bis zum Ablauf der Zeitüberschreitung. Mit der folgenden Anweisung wird der nichtgeblockte Modus eingestellt, das Skript läuft auch ohne eine Antwort weiter:

set_socket_blocking()

```
$mode = set_socket_blocking($fp, 0);
```

Das Abwarten einer Antwort erzwingt dagegen folgender Aufruf:

psockopen()

```
$mode = set_socket_blocking($fp, 1);
```

Problematisch ist auch die Belastung des Servers durch das ständige Öffnen und Schließen der Verbindungen. Auch wenn das explizite Schließen mit `fclose` am Ende des Skripts nicht erfolgt, wird PHP die Verbindungen schließen. Einen Ausweg bieten persistente Verbindungen. Dazu ersetzen Sie einfach `fsockopen` durch:

```
$fp = psockopen("server.tld", "80");
```

Fehlerbehandlung

Beide Socket-Funktionen, `fsockopen` und `fpsockopen`, können um Elemente der Fehlerbehandlung erweitert werden. Dies ist wichtig, weil gerade bei Internet-Verbindungen Fehler sehr häufig auftreten. Erweitern Sie den Funktionsaufruf folgendermaßen:

```
$fp = fsockopen("server.tld", "80", $errno, $errdescr);
```

Die Variable `$errno` enthält nun eine Fehlernummer, in `$errdescr` finden Sie eine von PHP generierte Beschreibung. Den weiteren Umgang mit dem Fehler müssen Sie nun wieder selbst programmieren.

Einen tieferen Einblick in die Netzwerkfunktionen, zu denen auch die Funktion `fsockopen` gehört, erhalten Sie in Kapitel 0 9

Fortgeschrittene Programmierung.

4.7.3 CURL-Funktionen

Die CURL-(Client Uniform Resource Locator)-Funktionen erlauben den direkten Zugriff auf Ressourcen anderer Server, die über das Internet erreichbar sind. Sie können Zugriff per GET oder POST simulieren, ohne die Spezifika des Protokolls HTTP zu kennen.

Funktionsübersicht

- `curl_close`

Schließt eine CURL-Sitzung und verwirft die eingestellten Parameter.

- `curl_errno`

Gibt die letzte Fehlernummer zurück. Wenn kein Fehler auftrat, wird 0 zurückgegeben.

- `curl_error`

Die Funktion gibt den Text der letzten Fehlermeldung zurück.

- `curl_exec`
Führt die Abfrage des Servers mit den voreingestellten Parametern aus.
- `curl_getinfo`
Mit dieser Funktion werden Informationen über den letzten Transfer ermittelt.
- `curl_init`
Diese Funktion initialisiert eine CURL-Sitzung und gibt ein Handle zurück, das die anderen CURL-Funktionen benötigen. Es wird dort als *curlhandle* bezeichnet.
- `curl_setopt`
Die Funktion setzt Optionen, die den bevorstehenden Zugriff auf eine URL näher beschreiben.
- `curl_version`
Die Funktion gibt die Version der verwendeten CURL-Bibliothek aus.

Eröffnen und Schließen einer CURL-Sitzung

Diese Funktion `curl_init` initialisiert eine CURL-Sitzung und gibt ein Handle zurück, das die anderen CURL-Funktionen benötigen. Es wird dort als *curlhandle* bezeichnet. Wenn Sie die URL angeben, wird die Option `CURLOPT_URL` gesetzt, es erfolgt aber noch kein Zugriff. `curl_close` schließt eine CURL-Sitzung und verwirft die eingestellten Parameter. Die folgenden Beispiele aktivieren vollständige Verbindungen und nutzen beide Funktionen.

`curl_errno` gibt die letzte Fehlernummer zurück. Wenn kein Fehler auftrat, wird 0 zurückgegeben. Ein Anwendungsbeispiel finden Sie nachfolgend. Die Funktion `curl_error` gibt den Text der letzten Fehlermeldung zurück.

```
<?php
$ch = curl_init("http://srv1.comzept-gmbh.dex");
curl_setopt ($ch, CURLOPT_HEADER, 1);
curl_exec ($ch);
if (curl_errno($ch) != 0) echo curl_error($ch);
curl_close ($ch);
?>
```

Hier ist offensichtlich die Domain falsch geschrieben (Toplevel *dex*), sodass die Namensauflösung fehlschlagen muss. Die Ausgabe zeigt, wie CURL darauf reagiert:

Starten:
`curl_init()`
Beenden:
`curl_close()`

Fehlerbehandlung:
`curl_errno()`
`curl_error()`

Listing 4.46:
curl_error: Nutzung
der Fehlerfunktionen

Abbildung 4.10:
Fehlerausgabe des
CURL-Moduls

```
Couldn't resolve host 'srv1.comzept-gmbh.dex'
```

Informationen über die letzte CURL-Sitzung ermitteln

Mit der Funktion `curl_getinfo` werden Informationen über den letzten Transfer ermittelt.

`curl_getinfo(int curlhandle);`

Das zurückgegebene assoziative Array enthält folgende Felder (in Klammern finden Sie Konstanten, die als zusätzliche Option ein Feld vorauswählen):

- `url` (`CURLINFO_EFFECTIVE_URL`)
Die URL, die tatsächlich verwendet wurde.
- `http_code` (`CURLINFO_HTTP_CODE`)
Der letzte HTTP-Statuscode, der zurückgegeben wurde.
- `header_size` (`CURLINFO_HEADER_SIZE`)
Ergibt die Größe des Headers.
- `request_size` (`CURLINFO_REQUEST_SIZE`)
Die Größe der gesamten Anforderung (Body).
- `total_time` (`CURLINFO_TOTAL_TIME`)
Die gesamte vergangene Zeit für den Vorgang.
- `namelookup_time` (`CURLINFO_NAMELOOKUP_TIME`)
Die Zeit für die Namensauflösung, abhängig von DNS-Servern.
- `connect_time` (`CURLINFO_CONNECT_TIME`)
Die Zeit, bis die Verbindung aufgebaut werden konnte.
- `pretransfer_time` (`CURLINFO_PRETRANSFER_TIME`)
Die Zeit vor dem Transfer (Reaktionszeit des Servers).
- `size_upload` (`CURLINFO_SIZE_UPLOAD`)
Die Größe des Uploads, wenn einer erfolgte.
- `size_download` (`CURLINFO_SIZE_DOWNLOAD`)
Größe des Downloads, wenn ein Body folgte.
- `speed_download` (`CURLINFO_SPEED_DOWNLOAD`)
Geschwindigkeit in Byte/Sekunde für den Download

- `speed_upload` (`CURLINFO_SPEED_UPLOAD`)

Geschwindigkeit in Byte/Sekunde für den Upload

Das folgende Beispiel zeigt die Anwendung ohne zusätzlichen Parameter:

```
<?php
$ch = curl_init("http://www.comzept-gmbh.de");
curl_setopt ($ch, CURLOPT_HEADER, 1);
curl_setopt ($ch, CURLOPT_TIMEOUT, 30);
curl_exec ($ch);
$arr = curl_getinfo($ch);
echo "<hr>";
foreach($arr as $field => $part) echo "$field => $part<br>";
curl_close ($ch);
?>
```

Listing 4.47:
Informationen über
die CURL-Sitzung

Die Ausgabe hängt von der konkreten Situation ab. In der Abbildung übermittelte der Server den Statuscode »302 Objekt verschoben«:

```
url => http://srv1.comzept-gmbh.de/book.sites
http_code => 302
header_size => 153
request_size => 141
filetime => 0
total_time => 0
namelookup_time => 0.01
connect_time => 0.03
pretransfer_time => 0.03
size_upload => 0
size_download => 0
speed_download => 0
speed_upload => 0
```

Abbildung 4.11:
Informationen über
die letzte
Anforderung

Der direkte Zugriff auf eine Option kann erfolgen, wenn ein zweiter Parameter angegeben wird, der der Funktion mitteilt, welcher Eintrag benötigt wird. Anstatt eines Arrays wird dann eine Zeichenkette zurückgegeben:

```
$code = curl_getinfo($ch, CURLINFO_HTTP_CODE);
echo $code;
```

Als Einschränkung können die Konstanten eingesetzt werden, die in der Optionsliste auf Seite 388 bereits gezeigt wurden.

Optionen für CURL-Funktionen einstellen

Die Funktion `curl_setopt` setzt Optionen, die den bevorstehenden Zugriff auf eine URL näher beschreiben.

```
curl_setopt($curlhandle, OPTION, $wert);
```

`OPTION` bezeichnet die Option, `$wert` den zu setzenden Wert. Im folgenden finden Sie eine Liste der zulässigen Optionen, die als Konstante angegeben werden:

- `CURLOPT_INFILESIZE`

Gibt die Größe einer Datei beim Upload an. Die Angabe erfolgt in Byte.

- `CURLOPT_VERBOSE`
Wenn der Wert `TRUE` ist, wird `CURL` alle Ereignisse ausgeben.
- `CURLOPT_HEADER`
Wenn `TRUE`, werden Header in die Ausgabe eingeschlossen.
- `CURLOPT_NOPROGRESS`
Wenn `TRUE`, wird keine Fortschrittsanzeige generiert. Dies ist die Standardeinstellung.
- `CURLOPT_NOBODY`
Wenn `TRUE`, wird der Body des Transfers nicht mit eingeschlossen.
- `CURLOPT_FAILONERROR`
Wenn `TRUE`, wird `PHP` den Transfer abbrechen, wenn ein `HTTP`-Fehler größer als 300 ausgegeben wird.
- `CURLOPT_UPLOAD`
Ist `TRUE`, wenn Dateien hochgeladen werden sollen.
- `CURLOPT_POST`
Wenn `TRUE`, wird `PHP` ein `HTTP-POST` ausführen, das der Server als Absender eines Formulars erkennt.
- `CURLOPT_FTPLISTONLY`
Wenn `PHP` bei einem `FTP`-Transfer die Dateien auflisten soll, ist dieser Wert auf `TRUE` zu setzen.
- `CURLOPT_FTPAPPEND`
Dateien werden angehängt, wenn die Option `TRUE` ist, sonst überschrieben.
- `CURLOPT_NETRC`
Wenn `TRUE`, versucht `PHP` die Datei `NETRC` zu lesen um Benutzernamen und Kennwort daraus zu entnehmen.
- `CURLOPT_FOLLOWLOCATION`
Wenn der Server eine Weiterleitung (Location-Header) sendet, kann `CURL` dem folgen. Dazu ist diese Option auf `TRUE` zu setzen.
- `CURLOPT_PUT`
Wenn `TRUE`, wird `HTTP-PUT` verwendet. `CURLOPT_INFILE` und `CURLOPT_INFILESIZE` müssen dazu auch gesetzt werden.

- `CURLOPT_MUTE`
Diese Option unterdrückt alle Meldungen.
- `CURLOPT_TIMEOUT`
Diese Option erwartet einen Wert in Sekunden, der die maximale Ausführungszeit bestimmt.
- `CURLOPT_LOW_SPEED_LIMIT`
Diese Option definiert eine Mindestübertragungsrate in Byte pro Sekunde, die erreicht werden muss, damit PHP den Transfer nicht abbricht.
- `CURLOPT_LOW_SPEED_TIME`
Diese Option bestimmt, wieviele Sekunden die mit `CURLOPT_LOW_SPEED_LIMIT` festgelegte Mindestübertragungszeit unterschritten werden muss, damit PHP abbricht.
- `CURLOPT_RESUME_FROM`
Diese Option bestimmt einen Offset in Bytes, ab dem der Transfer der Daten beginnt.
- `CURLOPT_SSLVERSION`
Kann 2 oder 3 sein und bestimmt damit die verwendete SSL-Version für verschlüsselte Verbindungen.
- `CURLOPT_TIMECONDITION`
Bestimmt die Behandlung der Option `CURLOPT_TIMEVALUE`. Kann `TIMECOND_IFMODSINCE` oder `TIMECOND_ISUNMODSINCE` annehmen. Der Einsatz ist nur für HTTP sinnvoll.
- `CURLOPT_TIMEVALUE`
Setzt einen Wert (als Unix-Timestamp), der angibt, ab wann Dokumente als geändert erkannt werden.
- `CURLOPT_URL`
Gibt die URL eines Dokuments, dessen Inhalt gelesen werden soll.
- `CURLOPT_USERPWD`
Gibt Benutzernamen und Kennwort für geschützte Zugriffe an. Die Form ist `username:password`.
- `CURLOPT_PROXYUSERPWD`
Gibt Benutzernamen und Kennwort für geschützte Zugriffe über einen Proxy an. Die Form ist `username:password`.

- `CURLOPT_RANGE`
Gibt einen Bereich an, der in der Form "A-B", "C-D" angegeben wird.
- `CURLOPT_POSTFIELDS`
Felder einer POST-Operation.
- `CURLOPT_REFERER`
Referer eines HTTP-Transfers.
- `CURLOPT_USERAGENT`
Setzt das Feld "user-agent".
- `CURLOPT_FTPPORT`
Adresse für FTP-POST, also eine IP-Nummer, Hostname usw.
- `CURLOPT_COOKIE`
Cookie-Informationen (Cookie-String).
- `CURLOPT_SSLCERT`
Dateiname eines Zertifikates.
- `CURLOPT_SSLCERTPASSWD`
Kennwort für das Zertifikat.
- `CURLOPT_COOKIEFILE`
Dateiname einer Datei mit Cookieinformationen im Netscape-Format (entspricht den HTTP-Headern).
- `CURLOPT_CUSTOMREQUEST`
Angabe eines anderen Kommandos außer GET oder HEAD bei HTTP-Anforderungen.
- `CURLOPT_FILE`
Dateihandle (mit `fopen` erzeugt), das auf eine Datei zeigt, in die die transferierten Daten platziert werden.
- `CURLOPT_INFILE`
Dateihandle (mit `fopen` erzeugt), das auf eine Datei zeigt, aus der die zu transferierenden Daten entnommen werden.
- `CURLOPT_WRITEHEADER`
Dateihandle (mit `fopen` erzeugt), das auf eine Datei zeigt, das die zu sendenden Header enthält.

- CURLOPT_STDERR

Dateihandle (mit fopen erzeugt), das auf eine Datei zeigt, in die Fehlermeldungen geschrieben werden.

- CURLOPT_RETURNTRANSFER

Veranlasst curl_exec statt des Status (TRUE oder FALSE) die empfangenen Daten zurückzugeben, sodass diese leichter verarbeitet werden können.

Die Funktion curl_version gibt die Version der verwendeten CURL-Bibliothek aus.

```
<?php
echo 'Eingesetzt wird CURL-Bibliothek: ';
echo curl_version();
?>
```

Listing 4.48:
curl_version: Versionsnummer des Moduls

Ein mögliche Ausgabe sehen Sie in der folgenden Abbildung:

```
Eingesetzt wird CURL-Bibliothek: libcurl 7.8 (OpenSSL 0.9.6a)
```

Abbildung 4.12:
Ausgabe der Curl-Version

4.7.4 FTP-Funktionen

Speziell für die Verbindung mit FTP-Servern können die FTP-Funktionen eingesetzt werden. Diese implementieren das Protokoll korrekt und sind flexibler und einfacher als der Wrapper für die fopen-Funktion.

Funktionsübersicht

Die folgende Funktionsliste verschafft einen Eindruck vom Umfang der Möglichkeiten:

- ftp_connect
Öffnet eine FTP-Verbindung.
- ftp_pwd
Gibt den Namen des aktuellen Verzeichnisses zurück.
- ftp_cdup
Wechselt in das übergeordnete Verzeichnis.
- ftp_mkdir
Erzeugt ein neues Verzeichnis.
- ftp_rmdir
Löscht ein Verzeichnis auf dem FTP-Server.

- `ftp_nlist`
Gibt eine Liste der Dateien im aktuellen Verzeichnis zurück.
- `ftp_systype`
Gibt den Systemcode des FTP-Servers zurück.
- `ftp_pasv`
Schaltet den passiven Mode ein oder aus.
- `ftp_get`
Lädt eine Datei vom FTP-Server.
- `ftp_fget`
Lädt eine Datei vom FTP-Server und speichert den Inhalt in einer offenen lokalen Datei.
- `ftp_put`
Sendet eine Datei zum FTP-Server.
- `ftp_fput`
Sendet aus einer lokalen geöffneten Datei Inhalte zum Server.
- `ftp_size`
Gibt die Größe einer Datei auf dem FTP-Server zurück.
- `ftp_mdtm`
Gibt den Zeitpunkt der letzten Änderung einer Datei auf dem FTP-Server zurück.
- `ftp_rename`
Benennt eine Datei auf dem FTP-Server um.
- `ftp_delete`
Löscht eine Datei auf dem FTP-Server.
- `ftp_quit`
Schließt eine FTP-Verbindung.

Für viele Funktionen finden Sie in diesem Abschnitt Anwendungsbeispiele.

An einem FTP-Server anmelden

Verbinden:
`ftp_connect()`

Die Anwendung erfolgt in ähnlicher Weise wie die Dateifunktionen. Die Funktion `ftp_connect` gibt ein Handle der Verbindung zurück. Mit diesem Handle arbeiten alle anderen Funktionen.

Die Funktion `ftp_connect` verbindet mit einem FTP-Server und gibt ein Handle auf diese Verbindung zurück, das von allen folgenden FTP-Funktionen genutzt wird.

Der optionale zweite Parameter *portnumber* gibt eine optionale Portnummer an. Erfolgt keine Angabe, wird der Standardport 21 benutzt.

```
<?php
$host = "srv1.comzept-gmbh.de";
$open = ftp_connect($host);
if ($open) echo "FTP-Server <b>$host</b> gefunden";
?>
```

Listing 4.49:
ftp_connect: Verbindung zu einem FTP-Server herstellen

Dieses Skript sollte folgendes ausgeben:

```
FTP-Verbindung
FTP-Server srv1.comzept-gmbh.de gefunden
```

Abbildung 4.13:
FTP-Verbindung erfolgreich eröffnet

Die Variable *\$open* dieses Beispiels enthält ein Handle auf die Verbindung, dass die andere FTP-Funktionen nutzen, um auf diese Verbindung zuzugreifen.

Die Funktion `ftp_login` führt eine Anmeldung an einem FTP-Server aus. Im Erfolgsfall wird `TRUE` zurückgegeben, sonst `FALSE`. Erst nach dem Ausführen der Anmeldung können viele FTP-Server verwendet werden.

Anmelden:
ftp_login()

```
<?php
$host = "srv1.comzept-gmbh.de";
$name = "Administrator";
$pass = "clemens";
$open = ftp_connect($host);
if ($open) {
    echo "Verbindung zu <b>$host</b> hergestellt.<br>";
    $logged = ftp_login($open, $name, $pass);
    if ($logged) {
        echo "Anmeldung erfolgt";
    }
}
?>
```

Listing 4.50:
ftp_logon:
An einem FTP-Server nicht anonym anmelden

Dieses Beispiel zeigt, wie die Verbindung komplett hergestellt wird. Alle anderen FTP-Funktionen benötigen diese Eröffnung der Verbindung. Die folgenden Beispiele basieren deshalb auf diesem Skript, ohne dass dies immer wieder aufgeführt wird.

Die Funktion `ftp_pwd` gibt den Namen des aktuellen Verzeichnisses zurück. War die Verbindung nicht möglich, wird `FALSE` zurückgegeben. Angewendet wird es in Listing 4.51.

ftp_pwd()

Die Funktion `ftp_chdir` wechselt das aktuelle Verzeichnis auf dem Server. War der Wechsel erfolgreich, wird `TRUE` zurückgegeben, sonst

ftp_chdir()

FALSE. Das Beispiel zeigt, wie Sie Laufzeitfehler beim Zugriff auf nicht vorhandene Verzeichnisse abfangen können:

Listing 4.51:
*ftp_chdir: Wechseln
des aktuellen
Verzeichnisses*

```
<?php
$host = "srv1.comzept-gmbh.de";
$name = "Administrator";
$pass = "clemens";
$open = ftp_connect($host);
ftp_login($open, $name, $pass);
@ftp_chdir($open, "wwwroot") or die("Fehler bei chdir");
echo ftp_pwd($open);
?>
```

ftp_cdup()

ftp_cdup wechselt eine Verzeichnisebene höher. Das folgende Beispiel zeigt, wie die Funktion verwendet wird.

Listing 4.52:
*ftp_cdup: Wechseln
des Verzeichnisses
eine Ebene höher*

```
<?php
$host = "srv1.comzept-gmbh.de";
$name = "Administrator";
$pass = "clemens";
$open = ftp_connect($host);
ftp_login($open, $name, $pass);
@ftp_chdir($open, "wwwroot") or die("Fehler bei chdir");
echo ftp_pwd($open);
echo "<br>";
ftp_cdup($open);
echo ftp_pwd($open);
?>
```

Ein weiteres Beispiel zeigt, wie das Erzeugen eines Verzeichnisses funktioniert:

Listing 4.53:
*ftp_mkdir: Erzeugen
eines Verzeichnisses*

```
<?php
$host = "srv1.comzept-gmbh.de";
$name = "Administrator";
$pass = "clemens";
$open = ftp_connect($host);
ftp_login($open, $name, $pass);
ftp_mkdir($open, "myupload");
ftp_chdir($open, "myupload");
echo "Aktuelles Verzeichnis: ". ftp_pwd($open);
?>
```

Die Ausgabe zeigt, dass das Verzeichnis erfolgreich erstellt wurde (anderfalls wäre ftp_chdir fehlgeschlagen):

Abbildung 4.14:
*Ausgabe des Skripts
aus Listing 4.53*

FTP-Verbindung
Aktuelles Verzeichnis: /myupload

ftp_rmdir löscht das angegebene Verzeichnis, welches leer sein muss. Es darf nicht mehr das aktuelle Verzeichnis sein. Der Name muss mit einem Schrägstrich beginnen.

Das folgende Beispiel zeigt die Anwendung:

```
<?php
$host = "srv1.comzept-gmbh.de";
$name = "Administrator";
$pass = "clemens";
$open = ftp_connect($host);
ftp_login($open, $name, $pass);
$d = @ftp_mkdir($open, "upload") or print("Bereits vorhanden");
ftp_chdir($open, "upload");
echo "Aktuelles Verzeichnis: ". ftp_pwd($open);
ftp_cdup($open);
ftp_rmdir($open, "/upload");
echo "<br>Aktuelles Verzeichnis: ". ftp_pwd($open);
?>
```

Listing 4.54:
*ftp_rmdir: Entfernen
eines Verzeichnisses*

Die ftp_cdup-Funktion stellt sicher, dass das Verzeichnis nicht mehr das aktuelle ist.

FTP-Verbindung

```
Bereits vorhanden
Aktuelles Verzeichnis: /myupload
Aktuelles Verzeichnis: /
```

Abbildung 4.15:
*Ausgabe des Skripts
aus Listing 4.54*

ftp_nlist listet die Dateien des angegebenen Verzeichnisses auf. Der Name muss mit einem Schrägstrich beginnen. Die Liste der Dateinamen wird als Array zurückgegeben. Im Fehlerfall wird FALSE zurückgegeben.

```
<?php
$host = "srv1.comzept-gmbh.de";
$name = "Administrator";
$pass = "clemens";
$open = ftp_connect($host);
ftp_login($open, $name, $pass);
@ftp_chdir($open, "/") or print("Fehler"); // Stammverzeichnis
echo "Verzeichnis: " . ftp_pwd($open);
echo "<hr noshade>";
$nlist = ftp_nlist($open, "/phpBookNew"); // Verzeichnisauswahl
foreach($nlist as $file) {
    if (@ftp_chdir($open, "/" . $file)) {
        echo "<DIR> ";
    } else {
        echo "<FILE> ";
    }
    echo basename($file) . "<br>";
}
?>
```

Listing 4.55:
*ftp_nlist: Dateiliste
auf einem FTP-
Server*

Die Ausgabe zeigt Verzeichnisse und Dateien an:

Abbildung 4.16:
Ausgabe von
Informationen eines
FTP-Servers

```

FTP-Verbindung
Verzeichnis: /
<DIR> applicat
<DIR> beispiel
<DIR> extension
<DIR> php_solutions
<DIR> shop
<DIR> support
<FILE> regpar.php3
<DIR> php4
<FILE> Kapitel111.html
<FILE> applicat.html
<FILE> beispiel.html
<FILE> blank.html
<FILE> extension.html
<FILE> frame2.html
<FILE> frame3.html
<FILE> frame3a.html
<FILE> frame4.html
<FILE> frame5.html
<FILE> frame6.html
<FILE> frame7.html
<FILE> frame8.html

```

**Ausführliche
Informationen:
ftp_rawlist()**

Die Funktion `ftp_rawlist` gibt eine detaillierte Liste von Dateien im angegebenen Verzeichnis zurück. Die Funktion führt intern das FTP-Kommando LIST aus. Das resultierende Array enthält die Ausgabe des LIST-Kommandos, wobei jedes Element eine Zeile enthält. Eine Bearbeitung erfolgt nicht, das Array enthält praktisch die Rohdaten. Jede Zeile enthält folgende Angaben (die Ziffer am Anfang bezeichnet die Spaltennummer):

- 1. d = Verzeichnis, - = Datei, l = Link
- 2. Zugriffsrechte (r = lesen, w = schreiben, x = ausführen); erste Gruppe: User, zweite Gruppe: Group, dritte Gruppe: Other
- 3. Betriebssystemabhängig, bei Unix entspricht es meist dem `ls -l` Kommando, bei Windows in der Regel gleich 1.
- 4. Eigentümer
- 5. Gruppe
- 6. Größe in Bytes
- 7. Monat
- 8. Tag
- 9. Jahr, wenn das aktuelle Jahr ein anderes ist, als das angezeigte, sonst die Uhrzeit
- 10. Datei- oder Verzeichnisname

Die Zerlegung der Zeilen muss selbst besorgt werden – am einfachsten lässt sich dafür `explode` einsetzen.

ftp_systype()

Die Funktion `ftp_systype` erkennt den Systemtyp des FTP-Servers. Bei Windows NT ist dieser Wert ebenso wie bei Window 2000 die Zeichenfolge »Windows_NT«.

```
<?php
$host = "srv1.comzept-gmbh.de";
$name = "Administrator";
$pass = "clemens";
$open = ftp_connect($host);
ftp_login($open, $name, $pass);
echo ftp_systype($open);
?>
```

Listing 4.56:
*ftp_systype: System
des FTP-Servers
erkennen*

`ftp_pasv` schaltet den passiven Modus *pasvmode* ein (TRUE) oder aus (FALSE). **ftp_pasv()**

```
ftp_pasv($fthdl, TRUE);
```

Normalerweise wird der Abruf von Daten durch einen Befehl auf dem Kommandoport 23 durch den Server initiiert. Manchmal funktioniert dies aber nicht, weil eine Firewall entsprechend konfiguriert ist und der Client dies nicht erkennt. Dann kann der passive Modus genutzt werden, wobei der Client Daten ohne gesondertes Kommando annimmt.

Übertragen von Dateien auf einen FTP-Server

Für die Übertragung von Daten stehen insgesamt vier Funktionen zur Verfügung. `ftp_fput` sendet Daten einer geöffneten Datei unter Angabe des Dateihandles, `ftp_put` dagegen akzeptiert einen Dateinamen. Beide Funktionen kennen einen weiteren Parameter, für den eine der beiden folgenden Konstanten angegeben werden:

ftp_put()
ftp_fput()

- `FTP_ASCII`
Übertragung als Textdatei.
- `FTP_BINARY`
Übertragung als Binärdatei.

Die Funktion überträgt nur den Teil der Datei ab der Position des internen Dateizeigers. Verwenden Sie zuvor `rewind`, um den Zeiger auf das erste Zeichen zu setzen.



`ftp_get` lädt eine Datei aus dem aktuellen Verzeichnis des FTP-Servers auf das lokale System, auf dem das PHP-Skript läuft – dies kann ebenso ein Server sein. Es kann auch ein anderer Pfad angegeben werden. Neben dem Quell- und dem Zieldateinamen kann auch hier eine der beiden bereits bei `ftp_put` Konstanten eingesetzt werden.

ftp_get()
ftp_fget()

Die Funktion überschreibt vorhandene Dateien gleichen Namens kommentarlos. Auch hier gibt es eine zweite Funktion `ftp_fget`, die statt eines Dateinamens das Handle einer bereits geöffneten Datei erwartet.

Listing 4.57:

ftp_getput: Anlegen einer Sicherheitskopie auf dem FTP-Server (die lokalen Pfade sind hier für Linux angegeben, unter Windows richten Sie eine Freigabe mit dem Namen »/tmp« für die temporären Daten ein)

```
<?php
$host = "www.joergkrause.de";
$name = "testphp4";
$pass = "forall";
$back = "backup";
$dir = "php/support";
$open = ftp_connect($host);
ftp_login($open, $name, $pass);
ftp_chdir($open, ".");
echo "Verzeichnis: " . ftp_pwd($open);
echo "<hr noshade>";
$rawlist = ftp_rawlist($open, $dir);
@ftp_mkdir($open, $back);
foreach($rawlist as $entry) {
    if (substr($entry, 0, 1) == "-") {
        $s = split("[\t]+", $entry);
        $file = $s[8];
        echo "Lade: $dir/$file ... ";
        ftp_get($open, "/tmp/$file", "$dir/$file", FTP_BINARY);
        echo "Schreibe: $dir/$back/$file<br>";
        ftp_put($open, "$dir/$back/$file", "/tmp/$file", FTP_BINARY);
        unlink("/tmp/$file");
    }
}
?>
```

Dieses Skript kopiert den Inhalt eines Verzeichnisses auf einem FTP-Server in ein temporäres lokales Verzeichnis und danach zurück in ein anderes Verzeichnis auf dem FTP-Server. Die temporären Kopien werden gelöscht. Die Ausgabe kontrolliert den Kopiervorgang:



Hinweis

Die genaue Syntax und weitere interessante Anwendungsbeispiele finden Sie im Referenzhandbuch: »PHP 4. Die Referenz«, das ebenfalls bei Carl Hanser erschienen ist.

4.8 Sicherheit

Ob eigener Webserver oder beim Provider genutzter Webspaces – über die Sicherheit muss man sich in jedem Fall Gedanken machen. Mit mehr eigener Technik steigt natürlich der Aufwand. Einen ersten Einblick in den Umfang der Arbeiten zeigt dieser Abschnitt.

4.8.1 Warum absichern?

Jeder Server sollte gesichert werden, auch wenn es unbedeutend erscheint.

Auch wenn Sie vielleicht denken, dass sich auf Ihrem Server keine wertvollen Daten befinden und sich sowieso niemand dafür interes-

siert, sollten Sie sich absichern! Es gibt eine Menge Gründe und Gelegenheiten, Störungen in fremden Systemen hervorzurufen. Nicht immer ist es böse Absicht, manchmal ist es auch nur ein Versehen oder pure Unkenntnis. Nur abgesicherte Systeme widerstehen den ständigen Angriffen aus dem Internet.

Eine Standardinstallation eines Webserver ist nicht sicher! Sie müssen Einstellungen von Hand vornehmen, um den Server abzusichern.



Wer greift einen Server eigentlich an? Hier eine Auswahl:

- *Die Scharlatane*

Unkenntnis, falsche Einschätzung des eigenen Vermögens oder falsch verstandene Neugier treibt viele Halbtechniker an die Computer und zu immer neuen Versuchen, mal einen Blick hinter die Kulissen zu werfen.

- *Die Spione*

Kundendaten sind ein reizvolles und wertvolles Gut. Wird es leicht gemacht, kann es sich lohnen, auch kleine Server zu knacken.

- *Die Vandalen*

Zerstörung als Sport – das gibt es auch im Internet. Nicht immer sind Hacker gutwillig; oft werden Daten einfach zerstört. Zwar kann man sich gegen den Verlust durch Sicherheitskopien schützen, aber Serverausfälle und Technikerstunden kosten viel Geld.

4.8.2 Grundlagen der Authentifizierung

Die Authentifizierung von Nutzern kann per Skript ausgeführt werden oder die Leistungen des Betriebssystems in Anspruch nehmen. Diese Methode ist sicherer, setzt aber Systemkenntnisse voraus.

Das UNIX-Rechtesystem

In ➤ Abschnitt 4.6 *Zugriff auf das Dateisystem* (ab Seite 350) wurde bereits die Auswertung der bestehenden Zugriffsrechte behandelt. PHP wurde speziell für den plattformübergreifenden Einsatz entworfen und entsprechend eingeschränkt waren die Einsatzmöglichkeiten bei den Zugriffsrechten. Wenn man jedoch Skripte speziell für Unix²⁰ entwirft, stehen die vollen Systemrechte zur Auswertung zur Verfügung.

Das Rechtesystem von Unix kennt drei Stufen:

²⁰ Hier und im Folgenden gilt dies uneingeschränkt auch für Linux.

- user
- group
- other

Jeder Stufe kann für jede Datei eines der folgenden Rechte zugewiesen werden:

- x
Executable, Ausführbar
- r
Readable, Lesbar
- w
Writeable, Schreibbar

Die Rechte können also wie folgt aufgeschrieben werden:

```
rwxrwxrwx
```

In diesem Fall hätten alle Nutzer, egal in welcher Gruppe, alle Rechte. Die interne Darstellung weist jedem Recht ein Bit eines 16-Bit-Wortes zu. Von rechts beginnend, hat das erste Bit (x) die Wertigkeit 1, das zweite (w) die Wertigkeit 2 und das dritte (r) die Wertigkeit 4. Alle drei Rechte zusammen ergeben die Wertigkeit 7 (1 + 2 + 4).

Die Darstellung erfolgt üblicherweise jedoch nicht in dezimaler oder hexadezimaler, sondern in oktaler²¹ Form. Die oben gezeigte Vergabe aller Rechte entspricht dann 777. Wenn Sie in Newsgroups oder Mailinglisten mitdiskutieren wollen, sollten Sie diese Form beherrschen. Ein paar Beispiele:

- r--r--r-- entspricht 444
- rwxr--r-- entspricht 744
- r--rw---- entspricht 460

In PHP werden Oktalzahlen mit einer vorangestellten Null geschrieben, also 0444, 0744 oder 0750. Natürlich können Sie immer die dezimale Entsprechung berechnen. Da die Zahlenbasis 8 ist, wird der »Zehner« (eigentlich ja der Achter) mit 8 multipliziert, der »Hunderter« ist korrekt ein »Vierundsechziger« und wird mit 64 multipliziert, aus 0444 wird dann (dezimal) 292.

²¹ Das Oktalsystem hat die Zahlenbasis 8.

Verzeichnisse werden analog Dateien behandelt und können über identische Rechte verfügen. Dateien innerhalb eines Verzeichnisses erben die Rechte, bis sie explizit überschrieben werden.

Um mit PHP Dateirechte gezielt zu vergeben, gibt es die Funktionen `chmod` und `chgrp`. Dabei gilt die Regel, dass jeder Nutzer nur Rechte vergeben kann, die gleich oder schlechter als seine eigenen sind. Entscheidend bei der Ausführung von Skripten ist also der Status, unter dem der Webnutzer die PHP-Skripte ausführt. Bei der Apache-Installation unter Linux übernimmt der Nutzer die User- und Gruppenrechte vom Apache. Webserver sollten generell mit einem speziellen Account versehen werden, beispielsweise *nobody*, dem dann soweit wie möglich eingeschränkte Rechte vergeben werden. Für die Ausführung von Skripten genügt es, wenn Leserechte bestehen. Beim Umgang mit den bereits beschriebenen Dateibefehlen werden auch Daten geschrieben. In diesem Fall muss *nobody* oder *other* Schreibrechte bekommen.

chmod()
chgrp()

```
<?php
$ok = chmod('muster.txt', 0666);
$fp = fopen('muster.txt', 'w');
fclose($fp);
echo "Status: $ok";
?>
```

Listing 4.58:
chmod:
Schreibrechte setzen

Experimentieren Sie in diesem Skript mit dem zweiten Parameter der `chmod`-Funktion herum. `chgrp` funktioniert analog und stellt Rechte für die aktuelle Gruppe ein.

Mit `chown` können Sie auch den Besitz einer Datei übernehmen, vorausgesetzt, sie gehört niemandem mit höheren Rechten.

chown()

Umgang mit dem IIS und Windows NT/2000

Windows NT/2000 ist mit der WIMP-Installation nach LAMP und WAMP die dritthäufigste Installation für PHP. Die Vergabe der Rechte ist jedoch bei PHP stark an Unix angelehnt. Unter Windows NT funktioniert dies nur eingeschränkt. Vermutlich haben die Entwickler alle Windows-Versionen, von 95 bis 2000, in einen Topf geworfen und den kleinsten gemeinsamen Nenner gebildet. Windows 9x/ME kennen tatsächlich keine Dateiattribute, die Nutzern zugeordnet werden können. Dateien können nur global schreibgeschützt werden. Genau so arbeitet auch die Funktion `chmod`. Sie können unter Windows mit der folgenden Funktion das Schreibschutz-Flag entfernen:



```
chmod("datei", 0777);
```

Folgendermaßen kann diese Flag dann wieder gesetzt werden:

```
chmod("datei", 0000);
```

chgrp ist ebenso wie chown unter Windows ohne Funktion. Gleichwohl verfügen zumindest Windows NT und Windows 2000 über die nötigen Schutzmechanismen im Dateisystem. Hier bleibt Ihnen nichts weiter übrig, als auf die Hausmittel zurückzugreifen, also das Einstellen der Rechte im Dateisystem und im Webserver mit der Management-Konsole.

**Zugriffsrechte
müssen im IIS und
im Dateisystem
eingestellt werden**

Zugriffsrechte für den Webserver

Wenn Sie sich die Zugriffsrechte aufmerksam anschauen, werden Sie bemerken, dass für die Nutzung als Webserver kaum komfortable Einstellungen möglich sind. Um beispielsweise Skripte ausführen zu können, müssten Sie dem Konto *IUSR_Machine* global die Rechte Ausführen und Lesen geben.

Die Verwaltung der Zugriffsrechte für die Webnutzer erfolgt deshalb im IIS selbst. Die Einteilung erfolgt in vier Gruppen:

- **ZUGANGSBERECHTIGUNGEN**

Diese gelten für die anonymen Nutzer des Webservers.

- **LESEN**

Das Lesen von Dateien ist erlaubt.

- **SCHREIBEN**

Das Schreiben von Dateien und Verzeichnissen ist erlaubt.

- **INHALTSFILTER**

Damit wird die Verwaltung des Inhalts der Verzeichnisse dieses Webs kontrolliert.

- **ZUGRIFF PROTOKOLLIEREN**

Zugriffe werden in der Protokolldatei des Webservers erfasst.

- **VERZEICHNIS DURCHSUCHEN ERLAUBT**

Wenn sich in dem Verzeichnis keine Standarddatei befindet und der Nutzer mit der URL keine Datei angegeben hat oder diese Datei nicht gefunden wurde, kann eine Liste des Verzeichnisinhaltes angezeigt werden, wenn diese Eigenschaft eingeschaltet wurde.

- **INDIZIEREN DES VERZEICHNISSES**

Das Verzeichnis wird indiziert, wenn der Index-Server es in einem seiner Kataloge aufgelistet hat.

- FRONTPAGE-WEB

Dieses Verzeichnis ist ein FrontPage-Web. Nutzer, die mit FrontPage 2000/2001 oder Visual InterDev arbeiten, können mit erweiterten Funktionen darauf zugreifen.

Für die praktische Arbeit reicht das noch nicht aus. Auch für Skripte und Programme gibt es Einstellungen, die sich auf die Sicherheit beziehen. Unter der Gruppe Einstellungen der Anwendung können Sie auch die Zugriffsrechte der Skripte kontrollieren. Dazu gehören:

- KEINE

Skripte und Programme werden nicht ausgeführt.

- SKRIPT

Nur Skripte werden ausgeführt.

- AUSFÜHREN

Ausführbare Programme und Skripten werden ausgeführt.

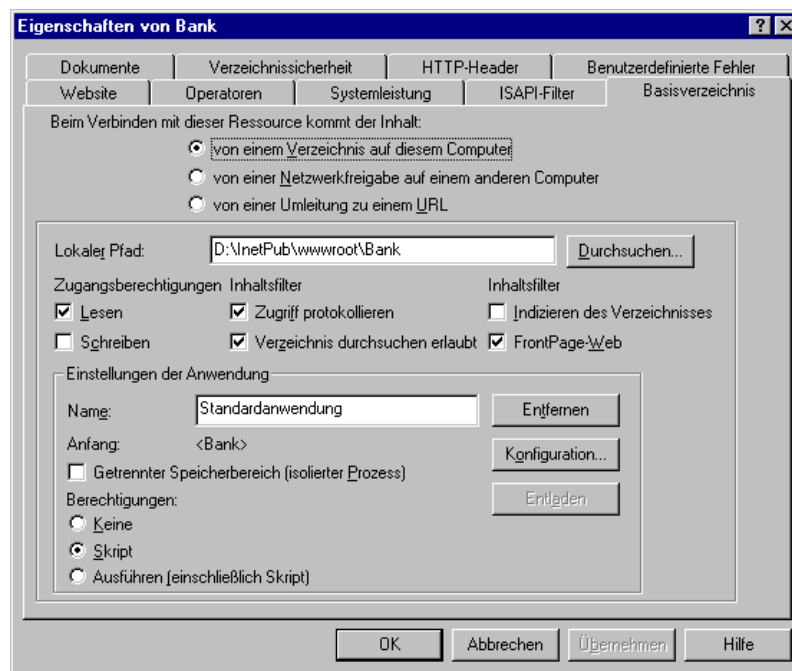


Abbildung 4.17: Einstellungen der Zugriffsrechte für den anonymen Zugriff im IIS 4 unter Windows NT. Der IIS 5 unter Windows 2000 arbeitet analog.

Welche Rechte stellen Sie nun in der Praxis ein? Auf jeden Fall sollten die Rechte für ein Webverzeichnis differenziert eingestellt werden. Wenn Sie mit PHP-Skripten arbeiten, macht es Sinn, diese in einem eigenen Verzeichnis zu speichern. Das Verzeichnis bekommt dann nur die Berechtigung SKRIPT. Deaktivieren Sie auch das Kontrollkästchen LESEN, denn der Nutzer muss und soll Skripte ja nicht lesen.

Die Rechte in der Praxis

Wenn Sie Textdateien zur Datenspeicherung einsetzen, dann speichern Sie die Dateien ebenso in einem eigenen Verzeichnis, wenn anonyme Nutzer darauf Schreibzugriff haben sollen. Geben Sie diesem Verzeichnis nur die Rechte SCHREIBEN und, wenn erforderlich, LESEN.

Tabelle 4.2:
Zugriffsrechte für
verschiedene
Dateitypen unter
Windows NT

Vorgang	Lesen	Schreiben	Skript	Ausführen
HTML-Datei	•			
PHP-Skript			•	
HTML und PHP	•		•	
Datenbank				
Textdatenbank	•	•		
Programm				•

Wenn Sie eine Datenbank in einem Verzeichnis des Webserverns ablegen, wird der Zugriff über den Datenbank-Server ausgeführt. In diesem Fall benötigen Sie keine Zugriffsrechte für anonyme Webnutzer. Sie können sowohl Schreiben als auch Lesen deaktivieren.

4.8.3 Seiten mit PHP schützen

Ein Zugriffsschutz für Bereiche einer Website wird häufig benötigt. Richtig interessant sind solche Lösungen, wenn die Daten für den Vergleich der Nutzernamen aus einer Datenbank kommen. Die Authentifizierung ist dabei aber der anspruchsvollere Teil, zumal einfache Lösungen auch schon mit den bisher gezeigten Mitteln auskommen.

Wie funktioniert die Authentifizierung?

Zuerst einmal muss klar sein, wie die Authentifizierung eines Webserverns abläuft. Das Geheimnis liegt im HTTP-Protokoll, das mit entsprechenden Kommandos auf geschützte Seiten hinweist. Eine ausführliche Beschreibung von HTTP finden Sie in ➡ Kapitel 9 *Fortgeschrittene Programmierung* ab Seite 629. Zum Verständnis der Authentifizierung soll ein kurzer Exkurs an dieser Stelle jedoch genügen.



Theorie

Fordert ein Browser eine geschützte Seite an, so antwortet der Webserver mit der Statusmeldung 403 – *geschützter Bereich*. Erst wenn diese Meldung empfangen wurde, sendet der Browser die ihm bekannten Nutzernamen und Kennwörter und versucht eine Autorisierung zu erreichen. Ist auch dieser Versuch erfolglos oder waren keine Kennwörter gespeichert, öffnet der Browser das bekannte Dialogfenster und fordert zur Eingabe von Nutzernamen und Kennwort auf. Der Webserver vergleicht die empfangenen Daten nun mit den Zugriffsrechten auf die Ressource und gibt diese im Erfolgsfall frei. Andernfalls lehnt er die Verbindung mit dem Status 401 ab.

Wenn Sie PHP als Modul in den Apache kompiliert haben, kann die Authentifizierung direkt vom PHP-Skript aus gesteuert werden. Im Sinne kompatibler Anwendungen ist dies aber nicht zu empfehlen.

Mit der CGI-Version (auch unter Linux) können Sie Seiten nur über das Dateisystem schützen. Nutzer, die so zugreifen möchten, müssen also als Nutzer im System explizit angelegt werden. Die Vergabe der Rechte unterscheidet sich nicht von der sonstigen Rechtevergabe und wurde bereits erwähnt. Der Grund liegt in der Arbeitsweise von CGI. Wenn Sie ein Verzeichnis oder eine Datei mit Hilfe von Einstellungen im Betriebssystem schützen, wird die Anforderung des Browsers an das betreffende Skript zurückgewiesen. Der Prozess, der davon betroffen ist, kann jedoch keine Header erzeugen oder andere HTTP-spezifische Aktionen veranlassen, sondern nur eine Fehlermeldung an die Standardausgabe STDOUT senden. Damit kann der Browser nichts anfangen. Für ihn ist der Vorgang beendet – Zugriff verweigert. Der Webserver wiederum sieht die Anforderung und liest eine Antwort, den Inhalt beurteilt er dabei nicht – der Vorgang wurde für ihn korrekt beendet. So kann man natürlich keine Authentifizierung aufbauen.

Grundlagen

Authentifizierung mit Apache-Modul

Wenn Sie die LAMP-Konfiguration verwenden und Apache als Modul einkompiliert haben, können Sie direkt auf die Header zugreifen. Eingriffe durch die CGI-Schnittstelle sind dann nicht störend. Der in [➡ Abschnitt Zugriffsrechte auf das Dateisystem](#) auf Seite 377 beschriebene Ablauf kann durch gezielte Ausgabe von HTTP-Headern gesteuert werden.

```
<?php
if(!isset($PHP_AUTH_USER)) {
    header("WWW-Authenticate: Basic realm=\"My Realm\"");
    header("HTTP/1.0 401 Unauthorized");
    $msg = "Zugriff nicht gestattet\n";
    echo $msg;
} else {
    $msg = "Hallo $PHP_AUTH_USER.<P>";
    $msg .= "Sie wurden mit Kennwort $PHP_AUTH_PW angemeldet.<P>";
}
?>
<html>
<body>
<h1>Sicherheit</h1>
<h3>Seiten schützen</H3>
<P>
<?php echo $msg; ?>
</body>
</html>
```

Listing 4.59:
authentication:
Authentifizierung
mit PHP

Das Beispiel in Listing 4.59 sendet dem Browser einen Header mit dem HTTP-Fehler 401. Daraufhin fordert der Browser vom Nutzer die Eingabe eines Namens und Kennwortes. Diese werden mit dem nächsten Request übermittelt und stehen in PHP als globale Variable zur Verfügung. Die Variable `$PHP_AUTH_USER` enthält den Anmeldenamen, `$PHP_AUTH_PW` das Kennwort (im Klartext!). Es obliegt dem Skript, Namen und Kennwort auszuwerten und beispielsweise mit einer vorhandenen Datenbank zu vergleichen.

Die Variable `realm` im Authentication-Header wird genutzt, um dem Browser eine Zuordnung der Server mit geschützten Seiten zu ermöglichen. Gespeicherte Kombinationen aus Nutzernamen und Kennwort werden immer einem sogenannten Realm (dt. *Gebiet, Reich*) zugeordnet, dem Server also. Wenn Sie in bestimmten Zeitabständen aus Sicherheitsgründen die erneute Eingabe des Anmeldenamens erzwingen möchten, ändern Sie den Parameter einfach. Der Browser kann dann die Zuordnung nicht mehr finden, sendet den alten Nutzernamen nicht aus und wird vom Skript zum Einblenden des Anmeldefensters gezwungen.



Wenn eine externe Authentifizierung genutzt wird, beispielsweise in Microsofts IIS oder über das Dateisystem in Linux, werden die Variablen `$PHP_AUTH_USER` und `$PHP_AUTH_PW` nicht gesetzt. Externe Nutzer, die sich anmelden, werden in `$REMOTE_USER` gespeichert. Die Verifizierung muss dann über das Benutzer- und Rechtesystem des Betriebssystems erfolgen.



Einen Ausweg für die Verwaltung großer Nutzergruppen ohne die Abhängigkeit von einer bestimmten Serverkonfiguration bietet eine Datenbanklösung auf Basis MySQL an. Dazu finden Sie in den folgenden Kapiteln viele Informationen.

Andere Authentifizierungsmethoden

Das Einblenden des Anmeldefensters ist sicher die eleganteste Methode zur Nutzerkontrolle. Sie können aber auch Formulare verwenden, den Inhalt auf der folgenden Seite mit einer Datenbank abgleichen und dann die weitere Verarbeitung von Skripten zulassen oder ablehnen. Das folgende Skript zeigt eine solche Vorgehensweise, wobei die registrierten Nutzer in einer Textdatei verwaltet werden.

Listing 4.60:
Einfache Abfrage von
Name und Kennwort
und Vergleich mit
den Einträgen in
einer Textdatei:
textauth.

```
<?php
$logok = FALSE;
if (isset($_login) and isset($_pass)) {
    $fp = fopen("passwd.pwd", "r");
    while ($line = fgets($fp)) {
        $arr = explode(";", $line);
        if (($arr[0] == $_login) and ($arr[1] == $_pass)) {
            $logok = TRUE;
        }
    }
}
```

```

        break;
    } /* end if */
} /* end while */
} /* end if */
if (!$logok) {
    echo "
    Bitte geben Sie Namen und Kennwort ein:<P>
    <FORM method=post action=\"$PHP_SELF\">
    Name: <INPUT type=text size=30 name=_login><BR>
    Kennwort: <INPUT type=text size=30 name=_pass>
    <input type=\"Submit\" name=\"submit\" value=\"Anmelden\">
    </FORM>
    <P>";
}
?>

```

Die Textdatei PASSWD.PWD hat einen einfachen Aufbau in der Form name,kennwort:

```

mueller,cv45vgs
haide,ccfgu
admin,admini3
baum33,enterthepass

```

Sie können damit beliebig viele Namen-/Kennwortkombinationen überwachen. Haben Sie erhöhte Sicherheitsansprüche, wäre es denkbar, die Einträge in der Textdatei mit der md5-Funktion zu verschlüsseln und bei der Prüfung entsprechend zu verfahren. Ein solcher Schutz ist durch normale Nutzer der Website kaum zu durchbrechen. Auch ein Diebstahl der Kennwortdatei wäre wertlos, da der Schlüssel nicht umkehrbar ist.

4.8.4 Sicherheitsmethoden der Webserver

Auch die Webserver selbst verfügen über Sicherheitsmechanismen. Für den Apache wird eine komfortable Lösung vorgestellt.

Verzeichnisschutz mit .htaccess

Die Auswahl, ob PHP als CGI-Programm oder als Apache-Modul läuft, wird oft nicht beim Entwickler liegen. Sie sollten sich also auf alle Varianten entsprechend vorbereiten und nur solche Skripte schreiben, die in allen Umgebungen laufen. Es ist unprofessionell, über die Einschränkungen bestimmter Systeme zu schimpfen und die eigene Lösung als Nonplusultra zu verkaufen.

Für den Apache-Webserver gibt es eine Lösung, wenn Sie Zugriff auf die .HTACCESS-Datei haben. Gehen Sie dazu folgendermaßen vor:

1. Erzeugen Sie eine leere Datei .HTPASSWD.

Für Apache und Linux

2. Geben Sie am Prompt Folgendes ein, wobei Sie <tag> durch die entsprechende Angabe ersetzen und {Aufforderung} eine Ausgabe des Betriebssystems darstellt:

```
$ htpasswd -c .htpasswd <nutzername>
{Password:} <kennwort>
{Password:} <kennwort>
```

3. Wiederholen Sie die Eingabe für jeden weiteren Nutzer ohne den Schalter -c (der steht für *create* und erzeugt die Datei).
4. Erzeugen Sie eine Datei mit dem Namen .HTACCESS und geben Sie Folgendes darin ein:

```
AuthAuthoritative Off
AuthName "Admin"
AuthType Basic
AuthUserFile /pfad_wo_immer_sie_es_haben/.htpasswd
require valid-user
```

Die Verwaltung der Nutzerzugriffe wird nun für das entsprechende Verzeichnis durch den Apache erfolgen. Wenn Sie eine ganz bestimmte Datei auf diesem Wege schützen möchten, können Sie folgenden Trick verwenden, der sich auch beim IIS bewährt hat.



Erzeugen Sie ein neues Verzeichnis, schützen Sie es wie beschrieben und legen Sie darin eine HTML-Datei (!) ab, die Folgendes enthält:

```
<META HTTP-EQUIV="refresh" content="0; url=/script/xyz.php">
```

Diese Datei ist geschützt, fordert die Eingabe von Nutzernamen und Kennwörtern, und leitet den Nutzer dann weiter auf das ungeschützte Skript.

**Die .htaccess
komfortabel mit
PHP erzeugen**

Wenn Ihr Provider Ihnen keinen Zugriff per Telnet erlaubt, können Sie den beschriebenen Weg nicht gehen. Ohne Konsole müssen Sie ein PHP-Skript verwenden. Ein Beispiel finden Sie in dem Projekt *htaccess*, das auf der CD und der Website zum Buch zu finden ist. Sie finden es auf der CD im Verzeichnis ANWENDUNGEN/HTACCESS. Die Skripte sind gut kommentiert und sollen hier nicht weiter vorgestellt werden.

Tipps für den Internet Information Server

Hier können Sie im IIS keine Nutzer verwalten. Sie müssen also eine Gruppe im Benutzermanager anlegen, die gewünschten Nutzer dort eintragen und den anonymen Zugriff im Webserver abschalten. Dazu gehen Sie folgendermaßen vor:

1. Starten Sie die Microsoft Managementkonsole.
2. Wählen Sie das betreffende Verzeichnis oder die Datei aus.



3. Wählen Sie im Dialog Eigenschaften die Registerkarte VERZEICHNISSICHERHEIT oder DATEISICHERHEIT.
4. Klicken Sie auf Bearbeiten im Abschnitt STEUERUNG DES ANONYMEN ZUGRIFFS...
5. Deaktivieren Sie das Kontrollkästchen ANONYMEN NUTZERN ZUGRIFF ERLAUBEN.

Aktivieren Sie das Kontrollkästchen STANDARD-AUTHENTIFIZIERUNG, deaktivieren Sie WINDOWS NT-HERAUSFORDERUNG.

4.9 Bilderzeugung

Bilder sind praktisch Bestandteil jeder Website. Diese nicht im statisch, mit einem Grafikprogramm, sondern erst beim Abruf der Seite per Skript zu erzeugen, ist eine der faszinierendsten Möglichkeiten in PHP.

4.9.1 Grundlegende Informationen

PHP besitzt eine sehr überzeugende und ungemein nutzbringende Eigenschaft – die dynamische Erzeugung von Bildern. Um Bilder erzeugen zu können, wird die GD-Bibliothek von Thomas Boutell benötigt. Die aktuelle Version liegt der PHP-Distribution bei. Außerdem lohnt ein Besuch der Website von Thomas Boutell:

<http://www.boutell.com/gd>

Ergänzend ist auch die Freetype-Bibliothek sinnvoll, womit TrueType-Fonts nutzbar werden. Abgesehen von den Windows-Fonts gibt es viele Server im Internet, die TrueType-Fonts aller Art anbieten. Zumeist sind diese auch für kommerzielle Zwecke kostenlos. So steht der dynamischen Erzeugung von guten Bildern nichts mehr im Weg.

Unter Windows müssen Sie die Erweiterung (DLL) in der Konfigurationsdatei PHP.INI freischalten:

```
extension=php_gd.dll
```

Der Windows-Distribution liegen verschiedene DLLs bei, die mit und ohne GIF-Unterstützung übersetzt wurden. Wenn Sie Ihr PHP bei einem Provider nutzen, der Linux einsetzt, müssen Sie mit dessen Angebot leben. Versionsnummer und Formatunterstützung ermittelt die Funktion `phpinfo`. In der Sektion GD finden Sie die entsprechenden Angaben.



Abbildung 4.18:
Unterstützung für
Bilderzeugung
ermitteln (aus der
Ausgabe von
phpinfo)

gd	
GD Support	enabled
GD Version	2.0 or higher
FreeType Support	enabled
FreeType Linkage	with freetype
JPG Support	enabled
PNG Support	enabled
WBMP Support	enabled

Mitte 2001 war die Version 2.0 aktuell.

Bildformate

GIF

Bilder, die von *jedem* Browser auch angezeigt werden können, müssen in den beiden Standardformaten des Webs, GIF oder JPEG vorliegen. Die Wahl des Formats hängt vom Zweck und Inhalt des Bildes ab. Die folgende Zusammenfassung enthält die wesentlichen Eigenschaften des Formates GIF:

- GIF-Bilder können maximal 256 Farben haben, allerdings wird eine Farbpalette mitgeladen, die sich aus einer größeren Farbmenge bedienen kann.
- Weniger Farben verringern die Größe der Datei erheblich.
- GIF komprimiert verlustfrei und eignet sich besonders für Schalter.
- GIF kennt das Attribut *Transparent*, mit dem sich interessante graphische Effekte erzielen lassen. Bildzellen, die transparent sind, lassen den Hintergrund durchscheinen. Nur so kann man beispielsweise runde Schalter simulieren.
- GIF kann Interlaced sein, das heißt, das Bild wird zeilenweise immer dichter werdend aufgebaut. Dies verkürzt subjektiv die Ladezeit.
- GIF-Bilder können animiert sein; mehrere Bilder in einer GIF-Datei laufen als kleiner Trickfilm ab.

JPEG

Als Alternative kann Joint Picture Expert Group (JPEG) verwendet werden. JPEG hat andere Eigenschaften und wird vor allem für Fotos und hochkomprimierte Bilder verwendet. Die wesentlichsten Merkmale sind:

- JPEG kann bis zu 16,7 Millionen Farben haben. Wenn der Monitor das nicht darstellen kann, wird der Browser durch Rasterung (Dithering) Zwischentöne bilden.

- Die Kompressionsrate ist einstellbar. Zunehmende Kompression führt allerdings zu unwiederbringlichen Qualitätsverlusten, denn JPEG nutzt ein verlustbehaftetes Kompressionsverfahren.

Sie können mit jedem Grafikprogramm Bilder für Ihre Webseiten erstellen, das diese beiden Formate gut exportiert. Sie finden Bilder auch im Web; achten Sie darauf, dass die Bilder ausdrücklich frei verwendbar sind. Und Sie können natürlich einen Grafiker mit der Erstellung von Bildern beauftragen und letztere dann in Ihre Programme und Webseiten einbinden.

Ein weiteres Format ist PNG, das verlustfrei und besser als GIF bei vergleichbarer Qualität komprimiert. Warum das eine Alternative sein könnte, lesen Sie im nächsten Abschnitt. Neben unbegrenzter Farbzahl werden auch Alpha-Kanäle (transparente bzw. durchscheinende Bereiche) unterstützt. PNG wird von allen aktuellen Browsern angezeigt.

Neuere GD-Bibliotheken unterstützen außerdem das Bitmap-Format WBMP, das unter anderem für Bilder in WAP-Handys zum Einsatz kommt.

Rechtsprobleme

Leider gestaltet sich derzeit die Zukunft des beliebten GIF-Formats recht schwierig. Die Fa. Unisys hält ein Patent auf den in GIF verwendeten Kompressionsalgorithmus LZW. Da die Nutzung mit nicht unerheblichen Lizenzgebühren belegt ist, wurde die GIF-Unterstützung aus der GD-Bibliothek entfernt. Immerhin handelt es sich um Freeware unter der GPL, sodass dem Entwickler kaum zugemutet werden kann, nun auch noch Lizenzgebühren zu entrichten.

Als bessere Alternative wird ohnehin PNG angesehen, das aber außer vom Internet Explorer erst ab Version 5 unterstützt wird. Der neue Netscape 6 Browser unterstützt PNG auch. Alternativ bleibt die alte Grafikbibliothek natürlich weiter verfügbar – zwar nicht offiziell über die Homepage von Thomas Boutell, aber über viele Server im Internet, wo sie ganz »zufällig« herumliegt.

4.9.2 Prinzipielle Arbeitsweise

Der Umgang mit Bildern ist recht einfach. Für den erfolgreichen Einsatz sollten Sie sich noch einmal die Funktionsweise von HTTP vor Augen führen. Jede Anforderung wird vom Server als ein in sich geschlossener, einzelner Prozess verstanden; das gilt eben auch für ein einzelnes Bild. Sie können also eine Anforderung so beantworten, dass der Browser nur ein Bild geliefert bekommt. Der Trick besteht nun in der Angabe einer entsprechenden Quelle:

```

```

PNG

Probleme mit GIF

Alternative PNG

Bilder aus PHP
heraus erzeugen

Dieses Tag fügt ein Bild in die Seite ein. Die Dateierweiterung interpretiert der Browser hier noch nicht. Er wartet ab, was der Server ihm liefert. Auf diese Anforderung hin wird nun auf dem Server ein Skript gestartet (*image.php*). Dieses Skript darf natürlich nur Bilddaten liefern, sonst nichts. Es ist deshalb sehr wichtig, dass innerhalb des Skripts nicht versehentlich Leerzeichen oder gar HTML ausgegeben werden. Damit wäre der Browser dann überfordert.

Die Festlegung der Art der übertragenen Datei erfolgt im Header. Entsprechend beginnt jedes »Bildskript« mit folgender Zeile – für das Format GIF:

```
<?php header("Content-type: image/gif"); ?>
```

Der folgende Abschnitt erläutert die wichtigsten Bildfunktionen und wie sie praktisch genutzt werden. Eine vollständige Erklärung der Funktionen finden Sie in der Referenz.

Für die Erzeugung von Bildern werden also im allgemeinen immer zwei Dateien benötigt: Eine zur Ausgabe der -Tags und eine weitere zur Erzeugung der Bilddaten.

Fonts

Bilder bestehen nicht nur aus Linien – öfter werden die Bildfunktionen eingesetzt, um Schrift zu erzeugen. Es stehen hier drei Möglichkeiten zur Verfügung:

- Eingebaute Fonts
- Type-1-Fonts (Postscript)
- Truetype-Fonts

Die eingebauten Fonts sind recht primitiv und wenig ansehnlich, eignen sich aber gut für erste Experimente, weil der Umgang damit einfacher als bei den anderen beiden Varianten ist. Type-1-Fonts sind weit verbreitet und werden beispielsweise mit dem Adobe Typemanager geliefert. Ebenso verbreitet ist Truetype, das Fontformat von Windows. Es gibt im Internet sehr viele Quellen für billige oder kostenlose Fonts, die für die in Bildern erreichbare Auflösung hinreichend gut sind. Sie sollten es vermeiden, kommerzielle Fonts von irgendwelchen Computern mitzunehmen und auf Webseite einzusetzen. Auch Fonts unterliegen Copyright-Bestimmungen.

Funktionsübersicht

Die folgende Übersicht zeigt die wichtigsten Funktionen auf einen Blick:

Bildfunktionen

- `getimagesize`

Ermittelt die Größe eines Bildes für alle Bildformate in Pixel.

- `imagealphablending`

Setzt die Stärke des Alpha-Kanals. Der Alpha-Kanal bestimmt die Durchsichtigkeit von Objekten im Bild.

- `imagearc`, `imagefilledarc`

Zeichnet einen Teil einer Ellipse. Die zweite Funktion füllt den Bereich anschließend mit einer Farbe.

- `imagechar`

Zeichnet ein Zeichen in horizontaler Richtung.

- `imagecharup`

Zeichnet ein Zeichen in vertikaler Richtung.

- `imagecolorallocate`, `imagecolordeallocate`

Weist dem Bild eine Farbe zu bzw. entfernt die Farbe wieder aus der Farbtabelle des Bildes. Die meisten Formate arbeiten mit dynamischen Farbtabellen, die nur die tatsächlich verwendeten Farben enthalten.

- `imagecolorat`

Ermittelt die Farbinformationen für ein bestimmtes Pixel des Bildes.

- `imagecolorclosest`

Diese Funktion ermittelt für eine Farbangabe die nächstliegende Farbinformation aus der aktuellen Farbtabelle des Bildes.

- `imagecolorclosestalpha`

Diese Funktion ermittelt für eine Farbangabe die nächstliegende Farbinformation aus der aktuellen Farbtabelle des Bildes und zusätzlich die entsprechenden Informationen über den Alpha-Kanal.

- `imagecolorexact`

Mit dieser Funktion wird der Index einer Farbe in der Farbtabelle ermittelt.

- `imagecolorexactalpha`

Mit dieser Funktion wird der Index einer Farbe in der Farbtabelle ermittelt und zusätzlich die entsprechenden Informationen über den Alpha-Kanal.

- `imagecolorresolve`

Mit dieser Funktion wird der Index einer Farbe in der Farbtabelle ermittelt. Wenn die Farbe in der Tabelle nicht vorhanden ist, wird die beste Alternative zurückgegeben.

- `imagecolorresolvealpha`

Mit dieser Funktion wird der Index einer Farbe in der Farbtabelle ermittelt. Wenn die Farbe in der Tabelle nicht vorhanden ist, wird die beste Alternative zurückgegeben. Zusätzlich wird die entsprechende Informationen über den Alpha-Kanal ermittelt.

- `imagegammacorrect`

Die Funktion stellt die Gamma-Korrektur (Farbkorrektur) ein.

- `imagecolorset`

Setzt gezielt eine bestimmte Farbe in der Farbtabelle eines Bildes.

- `imagecolorsforindex`

Ermittelt die Farbe in der Farbtabelle für einen bestimmten Index.

- `imagecolorstotal`

Ermittelt die Anzahl der Farben in der Farbtabelle.

- `imagecolortransparent`

Setzt eine Farbe als transparent. Dies gilt nur für Bildformate, die transparente Farben kennen.

- `imagecopy`

Kopiert einen Teil eines Bildes.

- `imagecopymerge`

Kopiert einen Bildteil und fügt es in ein anderes Bild ein.

- `imagecopymergegray`

Kopiert einen Bildteil und fügt es in ein anderes Bild ein. Dabei wird der kopierte Bildteil mit einer Graustufentabelle anstatt der Farbtabelle versehen.

- `imagecopyresized`

Kopiert und skaliert einen Bildteil.

- `imagecopyresampled`

Kopiert und skaliert einen Bildteil und berechnet dabei die Kompression neu.

- `imagecreate`

Erzeugt ein neues Bild. Das Bild ist erst leer und wird mit den anderen Funktionen »gefüllt«. Diese Funktion gibt ein Handle zurück, dass von den anderen Funktionen verwendet wird, um auf das neue Bild zuzugreifen.

- `imagecreatefromXXX`

Erzeugt ein Bild auf Basis eines vorhandenen. Anstatt XXX kann eines der unterstützten Bildformate gif, jpg, bmp oder png stehen. Außerdem kann ein, in einer Zeichenkette gespeichertes Bild, eingelesen werden (XXX wird dann durch string ersetzt).

- `imagedashedline`

Diese Funktion zeichnet eine gestrichelte Linie.

- `imagedestroy`

Zerstört das im Speicher aufgebaute Bild wieder.

- `imagefill`

Füllt einen geschlossenen Bereich mit einer Farbe.

- `imagefilledpolygon`

Mit dieser Funktion wird ein gefülltes Polygon gezeichnet.

- `imagefilledrectangle`

Die Funktion zeichnet ein gefülltes Rechteck.

- `imagefilltoborder`

Die Funktion füllt einen Bereich mit einer Farbe.

- `imagefontheight`, `imagefontwidth`

Ermittelt die aktuelle Fontgröße, getrennt nach Höhe und Breite.

- `imagegif`, `imagepng`, `imagejpg`, `imagebmp`

Gibt ein Bild an den Browser aus. Die Funktion bestimmt zugleich auch das Grafikformat. Die Funktionen existieren immer, es ist jedoch vom Typ der Bibliothek abhängig, welche Formate tatsächlich unterstützt werden.

- `imageinterlace`

Setzt oder löscht den Interlaced-Modus.

- `imageline`

Diese Funktion zeichnet eine Linie.

- `imageloadfont`
Setzt einen neuen Font im internen Format der GD-Bibliothek.
- `imagepolygon`
Zeichnet ein Polygon.
- `imagepsbbox`
Diese Funktion ermittelt die Maße, die ein Text in Type-1-Font in Anspruch nehmen würde.
- `imagepsencodefont`
Ändert die Kodierung eines Type-1-Fonts gemäß den Spezifikationen der Fonts. Dies gilt für Zeichen ab ASCII-Code 127, also den fremdsprachigen Teil. Damit lassen sich Standardfonts erweitern.
- `imagepsfreefont`
Gibt den von einem Type-1-Font verwendeten Speicher wieder frei.
- `imagepsloadfont`
Lädt einen Type-1-Font von der Festplatte in den Speicher, damit er von den anderen Font-Funktionen verwendet werden kann.
- `imagepsextendfont`
Erweitert oder verdichtet einen Type-1-Font anhand eines Parameters.
- `imagepsslantfont`
Stellt einen Type-1-Font schräg (kursiv). Die Schrägstellung ist frei wählbar.
- `imagepstext`
Zeichnet einen Text mit Type-1-Font in das Bild.
- `imagerectangle`
Diese Funktion zeichnet ein Rechteck.
- `imagesetpixel`
Mit dieser Funktion wird ein einzelnes Pixel gesetzt.
- `imagesetbrush`
Die Funktion setzt den Pinseltyp für die Linienfunktionen.
- `imagestring`
Schreibt eine Zeichenkette mit eingebauten Fonts horizontal.

- `imagestringup`
Schreibt eine Zeichenkette mit eingebauten Fonts vertikal.
- `imagesx`
Die Funktion ermittelt die Breite eines Bildes.
- `imagesy`
Die Funktion ermittelt die Höhe eines Bildes.
- `imagegetttbbox`
Ermittelt die Abmaße eines Textes mit TrueType-Font.
- `imagegettttext`
Schreibt einen Text mit einem TrueType-Font.
- `imagetypes`
Gibt die Bildtypen an, die tatsächlich unterstützt werden.
- `read_exif_data`
Liest den EXIF-Header eines JPG-Bildes.

Aufbau des Bilderzeugungsskripts

Die Erzeugung eines Bildes besteht immer aus mehreren Schritten. Zuerst legen Sie die Abmaße fest. Dann wird das »Rohbild« mit der Funktion `imagecreate` erzeugt. Diese Funktion gibt ein Handle zurück, dass die folgenden Funktionen verwenden, um Daten in dieses Bild zu schreiben. Dann werden die Farben für Vorder- und Hintergrund festgelegt. Jede Farbe erhält ebenfalls ein Handle auf die interne Farbtabelle des Bildes. Die erste festgelegte Farbe ist die Hintergrundfarbe. Nun werden die Objekte des Bildes erzeugt: Schrift, Linien, Kreise oder fertige Bilder von der Festplatte. Dabei werden alle Elemente sofort physisch in die Bilddatei (oder den Bildspeicher) geschrieben – Sie können den Vorgang nicht schrittweise rückgängig machen. Nun muss nur noch das Bild – das bislang im Speicher existiert – erzeugt werden. Die Funktion `imageXXX` sendet die Bilddaten direkt an den Browser, deshalb wird der entsprechende HTTP-Header mit den Informationen über den Bildtyp (MIME-Type) vorangestellt. Damit der Speicher nicht mit »Bildleichen« gefüllt wird, erfolgt anschließend die Zerstörung. Der Namensteil XXX der Funktion bestimmt, welcher Bildtyp erzeugt wird.

Einen ersten Eindruck dieses Ablaufs vermittelt das folgende Skript:

```
<?php
$breite = 200;
$hoehe  = 80;
```

Listing 4.61:
genimage: Erzeugen
von Bilddaten

Das Skript erzeugt dann die passenden Schaltflächen – hier im PNG-Format:

```
<?php
header( "Content-type: image/png");
if (!isset($s)) $s=11;
if (!isset($staticwidth)) {
    $dx=(imagefontwidth($font) * strlen($text));
} else {
    $dx=$staticwidth;
}
$dy=imagefontheight($font) + 6;
$ypad=10;
$ypad=10;
$im=imagecreate($dx+$ypad,$dy+$ypad);
if ($color== "blue") {
    $fgcolor=imagecolorallocate($im, 0,0,255);
} elseif ($color== "red") {
    $fgcolor=imagecolorallocate($im, 255,0,0);
} elseif ($color== "green") {
    $fgcolor=imagecolorallocate($im, 0,255,0);
} else {
    $fgcolor=imagecolorallocate($im,$color[0],$color[1],$color[2]);
}
$black=imagecolorallocate($im, 0,0,0);
$white=imagecolorallocate($im,255,255,255);
imagerecangle($im, 0, 0, $dx+$ypad-2, $dy+$ypad-2, $black);
imagerectangle($im, 0, 0, $dx+$ypad, $dy+$ypad, $white);
imagestring($im, $font, (int)($ypad*.80), $dy-(int)($ypad)-1,
    $text, $black);
imagepng($im);
imagedestroy($im);
?>
```

Listing 4.63:
genbutton:
Bilderzeugen (hier:
als PNG-Datei)

In bekannter Weise wird der Header erzeugt. Die Fontgröße wird in der Variablen `$s` übermittelt. Falls die Angabe fehlt, wird eine Standardgröße von 11 Punkt angenommen:

```
header("Content-type: image/png");
if (!isset($s)) $s=11;
```

Die Breite kann ebenso optional festgelegt werden. Erfolgt keine Angabe, wird die Breite des Bildes anhand des darzustellenden Textes ermittelt (`$staticwidth = 0`).

```
if (!isset($staticwidth)) $staticwidth=0;
```

Das Array `$size` enthält die mutmaßliche Größe des Bildes für den angegebenen Font, ermittelt mit der Funktion `imagettfbbox`:

```
$size = imagettfbbox($s, 0, "TIMES.TTF", $text);
```

Die Maße des Bildes werden nun daraus berechnet.

Wie es funktioniert

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

```
$dx = abs($size[2] - $size[0]);
$dy = abs($size[5] - $size[3]);
```

Alternativ kann die Größe auch direkt aus dem Font ermittelt werden:

```
if ($staticwidth==0){
    $dx = (imagefontwidth($font) * strlen($text));
} else {
    $dx = $staticwidth;
}
$dy=imageFontHeight($font);
```

Nun wird die Umrandung festgelegt:

```
$xpad=9;
$ypad=9;
```

Und das Bild kann (vorerst intern) erzeugt werden:

```
$im = imagecreate($dx+$xpad,$dy+$ypad);
```

Nun werden die Farben festgelegt:

```
if ($color == "blue") {
    $blue = imagecolorallocate($im, 0x3C,0x6D,0xAF);
} elseif ($color == "red") {
    $red = imagecolorallocate($im, 255, 0, 0);
} elseif ($color == "green") {
    $green = imagecolorallocate($im, 0, 255, 0);
} else {
    $blue = imagecolorallocate($im, 0, 0, 255);
}
```

Jetzt werden zwei Rechtecke erzeugt, die so platziert werden, dass ein Schatten für den 3D-Effekt entsteht:

```
imagerectangle($im, 0, 0, $dx+$xpad-1, $dy+$ypad-1, $black);
imagerectangle($im, 0, 0, $dx+$xpad, $dy+$ypad, $white);
```

Die nächste Funktion setzt den Text in den Schalter:

```
imagestring($im, $font, (int)($xpad*.80),
    $dy-(int)($ypad)-1, $text, $white);
```

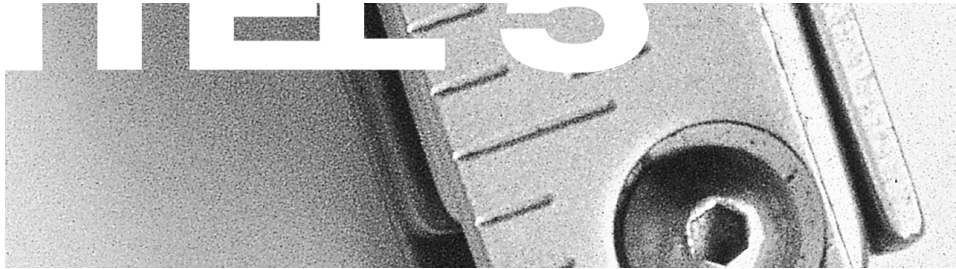
Das Bild ist nun fertig. Der folgende Befehl sendet es an den Browser:

```
imagepng($im);
```

Wenn Ihre Bibliothek GIF unterstützt, können Sie auch ImageGif einsetzen.

Zuletzt wird die Bildkopie im Speicher zerstört:

```
imagedestroy($im);
```



Praxis I – Lösungen für den Alltag

In diesem Kapitel finden Sie Skripte, die sofort eingesetzt werden können und häufig auftretende Probleme lösen. Es wird auch erläutert, wie die Skripte funktionieren und auf welchen Grundfunktionen sie aufbauen.

Lösung I: Mathematische Funktionen (Seite 425)

Lösung II: Dateioperationen (Seite 431)

Lösung III: Counter (Seite 439)

Lösung IV: E-Mail-Abruf (Seite 443)

Lösung V: Skripte für die Website (Seite 448)

5.1 Lösung I: Mathematische Funktionen

Mit kleinen Bibliotheken lässt sich der Programmieralltag erleichtern. Pflegen Sie solche Dateien und stellen Sie sich daraus eigene Sammlungen zusammen.

Einführung

Die folgenden Funktionserweiterungen sind ausschließlich mit den Mitteln aus PHP selbst entstanden und können unproblematisch in jedem Skript eingesetzt werden. Wenn Sie die hier vorgestellten Erweiterungen häufiger benötigen, nutzen Sie die Einbindung der Dateien mit `require`:

```
require("datei.inc.php");
```

Jeder Abschnitt enthält ein Thema, die Skripte sind alle auf der Buch-CD und in der aktuellsten Version im Internet zu finden.

5.1.1 Logarithmus mit unterschiedlichen Basen

Folgende Funktionen werden eingesetzt:

- `log10` (Es kann jede Logarithmusfunktion verwendet werden)

`log10()`

```
<?php
function baselog($base, $val) {
    return ((log10($val))/(log10($base)));
}
?>
```

Listing 5.1:
Funktion baselog

Wenn Sie als Basis e einsetzen möchten, gehen Sie folgendermaßen vor:

```
<?php
$var1 = 1.2847;
echo "$var1 = " . baselog(M_E, $var1);
?>
```



5.1.2 Fakultät

Für die Berechnung der Fakultät gibt es viele Lösungsansätze. Oft wird diese Funktion zur Demonstration der Rekursion eingesetzt:

```
<?php
function facultaet($n) {
    if ($n == 1) return 1;
    return $n * facultaet(--$n);
}
?>
```

Listing 5.2:
Funktion factorial

5.1.3 Größter gemeinsamer Teiler

Für viele mathematische Operationen ist der größte gemeinsame Teiler ein Basiswert. Die folgende Funktion berechnet diesen Wert.

Listing 5.3:
Funktion gcd

```
<?php
function gcd($x, $y) {
    $x = abs($x);
    $y = abs($y);
    if ( $x + $y == 0 ) { // 0 nicht sinnvoll
        return "FEHLER";
    } else {
        while ($x > 0) {
            $z = $x;
            $x = $y % $x;
            $y = $z;
        }
        return $z;
    }
}
?>
```

Das Beispiel zeigt, wie Sie diese Funktion nutzen können:

Listing 5.4:
Beispiel zum Testen
der Funktion im
Skript gcd

```
<?php
if (($result = gcd($var1, $var2)) <> "FEHLER") {
    echo "<p>Größter gemeinsamer Teiler von  

    <b>$var1</b> und <b>$var2</b> ist: $result";
} else {
    echo "<p>Eingabewerte ungültig.";
}
?>
```

5.1.4 Finanzmathematische Funktionen

Das folgende Skript berechnet den Verlauf eines Annuitätendarlehens. Es ist relativ einfach, kann aber leicht um Funktionen wie Disagio etc. ergänzt werden. Sie können es auf Ihrer Website direkt einsetzen, wenn Sie Kreditberechnungen anbieten möchten.

Listing 5.5:
Darlehens-
berechnung: finanz

```
<?php
function eingabewerte_pruefen($kredithoehe, $zins_nominal,
                               $laufzeit)
{
    return is_numeric($kredithoehe) && is_numeric($zins_nominal)
        && is_numeric($laufzeit);
}

function plan_fehlermeldung( )
{
    echo "Prüfen Sie bitte Ihre Eingaben.";
}
```

```

function plan_berechnen($kredithoehe, $zins_nominal, $laufzeit)
{
    $aufwand_gesamt = 0;
    $zins_gesamt    = 0;
    $tilgung_gesamt = 0;
    $tilgung_monat  = 0;
    $zins_monat     = 0;
    $zins_jahr      = 0;
    $tilgung_jahr   = 0;
    $aufwand_jahr   = 0;
    $tilgungssatz   = 100 / $laufzeit; // Volle Tilgung
    $ende           = FALSE;
    $restschuld     = $kredithoehe;
    $aufwand_monat_standard = $kredithoehe *
        ($zins_nominal + $tilgungssatz) / 100 / 12;
    $aufwand_monat = $aufwand_monat_standard;
    echo <<<HEADER
        <h4>Finanzierungsplan</h4>
        <table>
            <tr><td>Kreditheute</td><td>$kredithoehe DM</td>
            <tr><td>Nominalzins</td><td>$zins_nominal %</td>
            <tr><td>Laufzeit</td><td>$laufzeit Jahre</td>
        </table>
        <p>
        <table border=6 bgcolor=#FFE08>
            <tr><th>Jahr</th><th>Monat</th>
                <th width=100>Restschuld</th>
                <th width=100>Zins</th>
                <th width=100>Tilgung</th>
                <th width=100>Rate</th>
        HEADER;
        for ($jahr = 1; $jahr <= $laufzeit && !$ende; $jahr++) {
            $zins_jahr = 0;
            $tilgung_jahr = 0;
            $aufwand_jahr = 0;
            for ($monat = 1; $monat <= 12 && !$ende; $monat++)
            {
                $tilgung_monat = ($aufwand_monat - $restschuld *
                    $zins_nominal / 100 / 12 ) /
                    (( 12 - $zins_nominal / 100 ) / 12 );
                if ($restschuld <= $tilgung_monat)
                {
                    $tilgung_monat = $restschuld;
                    $zins_monat = $restschuld = 0;
                    $aufwand_monat = $tilgung_monat;
                    $ende = TRUE;
                } else {
                    $zins_monat = $aufwand_monat_standard - $tilgung_monat;
                    $restschuld -= $tilgung_monat;
                }
            }
        }
    }
}

```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

```

printf("<TR><TD><TD align=right>%d</TD>
      <TD align=right>%0.2f</TD>
      <TD align=right>%0.2f</TD>
      <TD align=right>%0.2f</TD>
      <TD align=right>%0.2f</TD>", $monat,
      $restschuld, $zins_monat, $tilgung_monat,
      $aufwand_monat);
$zins_gesamt += $zins_monat;
$tilgung_jahr += $tilgung_monat;
$zins_jahr += $zins_monat;
$aufwand_jahr += $aufwand_monat;
$tilgung_gesamt = $tilgung_jahr + $tilgung_monat;
$aufwand_gesamt += $aufwand_monat;
}
printf("<TR bgcolor=white>
      <TD align=left colspan=2>%d</TD>
      <TD align=right>%0.2f</TD>
      <TD align=right>%0.2f</TD>
      <TD align=right>%0.2f</TD>
      <TD align=right>%0.2f</TD>",
      $jahr, $restschuld, $zins_jahr, $tilgung_jahr,
      $aufwand_jahr);
}
printf("<TR><TD>Gesamt<TD align=right><b>%0.2f</b></TD></b>
      <TD align=right><b>%0.2f</b></TD>
      <TD align=right><b>%0.2f</b></TD>
      <TD align=right><b>%0.2f</b></TD>",
      $restschuld, $zins_gesamt,
      $tilgung_gesamt, $aufwand_gesamt);
echo "</TABLE>";
echo "<P><HR><EM>F&uuml;r die Richtigkeit dieser Angaben wird
      keine Gew&uuml;hr &uuml;bernommen.</EM><P>";
}
function plan ($kredithoehe, $zins_nominal, $laufzeit )
{
  if (eingabewerte_pruefen($kredithoehe,$zins_nominal,$laufzeit))
  {
    $zins_nominal *= 100;
    plan_berechnen($kredithoehe, $zins_nominal,
                  (int) $laufzeit);
  } else {
    plan_fehlermeldung();
  }
}
// -->
?>
<BODY>
<h1>Kreditberechnung</h1>
<h3>Geben Sie hier die vorhandenen Daten an:</h3>
<FORM ACTION="<?php echo $PHP_SELF ?>" METHOD="POST">
<TABLE>

```

```

<TR>
  <TD>Kreditheute</TD>
  <TD><INPUT NAME="kreditheute" TYPE="Text"
    VALUE="<?php echo $kreditheute ?>">
  </TD>
  <TD>DM</TD>
</TR>
<TR>
  <TD>Laufzeit</TD>
  <TD><INPUT NAME="laufzeit" TYPE="Text"
    VALUE="<?php echo $laufzeit ?>">
  </TD>
  <TD>Jahre</TD>
</TR>
<TR>
  <TD>Nominalzins</TD>
  <TD><INPUT NAME="zins_nominal" TYPE="Text"
    VALUE="<?php echo $zins_nominal ?>">
  </TD>
  <TD>%</TD>
</TR>
</TABLE>
<INPUT TYPE="reset" VALUE="L\u00f6schen">
<INPUT TYPE="submit" VALUE="Berechnen">
</FORM>
<h3>Berechnungsergebnis</h3>
<?php
$zins_nominal /= 100;
plan ($kreditheute, $zins_nominal, $laufzeit);
?>
</BODY>
</HTML>

```

Im Skript werden einige Annahmen getroffen, die durch weitere Abfragen im Formular gesteuert werden können. So wird beispielsweise die Tilgung gleichmäßig verteilt. Sie können die Variable *\$tilgungssatz* aber auch auf einen festen Wert setzen und dann die Schleife bis zum Ende laufen lassen. Im Ergebnis wird dann die Laufzeit berechnet, die notwendig ist, damit das Darlehen getilgt wird. Die Tilgungsdauer kann also geringer sein, als in der Jahresvorgabe erwartet wird.

Abbildung 5.1:
Anforderungen einer
Darlehens-
berechnung

Kreditberechnung

Geben Sie hier die vorhandenen Daten an:

Kredithöhe DM

Laufzeit Jahre

Nominalzins %

Fertig Lokales Intranet

Das Ergebnis der Berechnung umfasst eine Tabelle mit den monatlichen Raten, der Restschuld sowie dem Zins- und Tilgungsanteil.

Abbildung 5.2:
Teil des
Tilgungsplanes

Finanzierungsplan

Kredithöhe 50000 DM
Nominalzins 8 %
Laufzeit 5 Jahre

Jahr	Monat	Restschuld	Zins	Tilgung	Rate
	1	49161.07	327.74	838.93	1166.67
	2	48316.52	322.11	844.56	1166.67
	3	47466.29	316.44	850.22	1166.67
	4	46610.36	310.74	855.93	1166.67
	5	45748.69	304.99	861.68	1166.67
	6	44881.23	299.21	867.46	1166.67
	7	44007.95	293.39	873.28	1166.67
	8	43128.81	287.53	879.14	1166.67
	9	42243.76	281.63	885.04	1166.67
	10	41352.78	275.69	890.98	1166.67
	11	40455.82	269.71	896.96	1166.67
	12	39552.84	263.69	902.98	1166.67
1		39552.84	3552.84	10447.16	14000.00
	1	38643.80	257.63	909.04	1166.67
	2	37728.66	251.52	915.14	1166.67
	3	36807.37	245.38	921.28	1166.67
	4	35879.91	239.20	927.47	1166.67
	5	34946.21	232.97	933.69	1166.67
	6	34006.26	226.71	939.96	1166.67
	7	33059.99	220.40	946.27	1166.67
	8	32107.37	214.05	952.62	1166.67
	9	31148.36	207.66	959.01	1166.67
	10	30182.91	201.22	965.45	1166.67
	11	29210.99	194.74	971.93	1166.67
	12	28232.54	188.22	978.45	1166.67
2		28232.54	2679.70	11320.30	14000.00

Fertig Lokales Intranet

5.2 Lösung II: Dateioperationen

Der Dateiupload ist eine interessante Funktion. Das Skript zeigt, wie dies besonders komfortabel erfolgen kann.

5.2.1 Ersetzen in Dateien

Das mehrfache Ersetzen von Werten innerhalb einer Zeichenkette ist kein Problem, dafür stehen mehrere Funktionen zur Auswahl. Wenn der Ersetzungsvorgang in einer Datei vorgenommen werden soll, sieht die Sache schon anders aus. Das folgende Beispiel zeigt, dass es auch dafür eine einfache und kompakte Lösung gibt.

Ersetzen mit regulären Ausdrücken

Die Funktion kann problemlos überall eingebaut werden, wo eine derartige Leistung erforderlich ist. Typisch ist die Anwendung zum dauerhaften »verdichten« von HTML-Dateien durch entfernen von Leerzeichen und Tabulatoren. Dazu muss der passende reguläre Ausdruck als erster Parameter übergeben werden.

```
<?php
function massreplace($string1, $string2, $filename)
{
    $fp = fopen($filename, "r");
    $size = filesize($filename);
    $contents = fread($fp, $size);
    fclose($fp);
    $massreplace = preg_replace("/$string1/", $string2, $contents);
    $fp = fopen($filename, "w");
    fputs($fp, $massreplace);
    fclose($fp);
}
?>
```

*Listing 5.6:
Mehrfaches Ersetzen
in einer Datei:
massreplace
(Ausschnitt)*

Sie rufen die Funktion mit drei Parametern auf; *\$string1* ist die zu suchende Zeichenkette, *\$string2* die Ersatzzeichenkette und *\$filename* die Datei, in der gesucht werden soll.

Wie es funktioniert

Zuerst wird die Datei normal geöffnet:

```
$fp = fopen($filename, "r");
```

Dann wird die Größe der Datei ermittelt:

```
$size = filesize($filename);
```

Nun wird die gesamte Datei mit `fread` in eine Variable eingelesen. Das ist, gegenüber den bekannten Schleifen, deutlich einfacher und vor allem schneller.

```
$contents = fread($fp, filesize($filename));
```


Jetzt wird die Datei geschlossen und der Ersetzungsvorgang ausgeführt:

```
fclose($fd);
$massreplace = ereg_replace($string1, $string2, $contents);
```

Dann wird die veränderte Datei mit `fputs` wieder zurückgeschrieben:

```
$fp = fopen($filename, "w");
fputs($fp, $massreplace);
fclose($fp);
```

Da die Ersetzung auf einer Standardfunktion aufbaut (`preg_replace`), können Sie hier leicht eigene Konstruktionen einsetzen.

5.2.2 Umgang mit Verzeichnissen

Der Umgang mit Verzeichnissen ist nicht ganz einfach, wenn man die vorhandenen Funktionen direkt anwendet. Es fehlt beispielsweise eine Funktion, die den Inhalt eines Verzeichnisses in ein Array überführt, was eine weitaus komfortablere Form der weiteren Verarbeitung erlauben würde. Die folgende Lösung zeigt eine solche Funktion und verwendet sie, um eine Struktur auf einem Laufwerk rekursiv anzuzeigen.

*Listing 5.7:
dirtoarray: So
werden Verzeichnisse
in ein Array gelesen*

```
<?php
function DirToArray($sPath) {
    global $depth;
    $handle = opendir($sPath);
    while ($arrDir[] = readdir($handle)) {}
    closedir($handle);
    sort($arrDir);
    foreach($arrDir as $file) {
        if (!preg_match("/^\./", $file) and strlen($file)) {
            if (is_dir($sPath."/".$file)) {
                echo str_repeat("-", $depth);
                echo " <&lt;$file&gt;><br>";
                $depth++;
                DirToArray($sPath."/".$file);
                $depth--;
            } else {
                echo str_repeat("-", $depth);
                echo " $file";
            }
            echo "<br>";
        } /* end if */
    } /* end foreach */
    return $arrDir;
}
$depth = 1;
$path = "."; # Startpunkt
echo <<<DIR
```

```

    <h3>Ausgabe eines Verzeichnisbaumes</h3>
    <pre>
DIR;
$arrDir = DirToArray($path);
echo ' </pre>';
?>

```

Diese Lösung ist auch eine andere Form der Darstellung von Verzeichnisstrukturen. Im Kern sind nur wenige Befehle für die Übertragung der Verzeichnisse in ein Array verantwortlich. Mit der folgenden Zeile wird ein Verzeichnis (ohne Unterverzeichnisse) gelesen:

Wie es funktioniert

```
while ($arrDir[] = readdir($handle)) {}
```

Auf dieses Array können alle Sortierfunktionen angewendet werden:

```
sort($arrDir);
```

Auch die klassischen Funktionen zum Lesen des Arrays funktionieren:

```
foreach($arrDir as $file) {
```

Der übrige Teil des Skripts nutzt die Technik der Rekursion, um beliebige Verzeichnisstrukturen zu lesen. In einer globalen Variablen *\$depth* wird die Verschachtelungstiefe mitgeführt:

```

$depth++;
DirToArray($sPath."/".$file);
$depth--;

```

5.2.3 Universeller Dateiupload

Das folgende Skript bedient beliebige Formulare, mit denen Dateien hochgeladen werden können. Dabei werden zusätzliche Informationen in versteckten Feldern übergeben. So kann beim Entwurf des HTML-Formulars entschieden werden, wie sich das Hochladen auswirkt.

Zuerst wird das Formular vorgestellt. Es ruft sich selbst auf, deshalb wird die Klasse *upload* hier eingebunden.

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<?php require('uploadclass.php4'); ?>
<?php
function brecho($str)
{
    print strlen($str) > 0 ? $str . '<br>' : '';
}
?>
<html>
<head>
    <title>Upload</title>
</head>

```

Listing 5.8:
Skript zur Demonstration der Klasse *uploadclass* (siehe nächstes Listing)

```

<body>
<div style="font-family:Verdana">
<h1>Hochladen von Dateien</h1>
<form method="post" enctype="multipart/form-data"
  action="<?php echo $PHP_SELF ?>">
<table border="0">
  <tr>
    <td>
      Vollbild, JPG, maximal 50 KByte<br>
      <input type="Hidden" name="UPLOADDIR_bild1"
        value="user1">
      <input type="Hidden" name="UPLOADMIME_bild1"
        value="image/jpg">
      <input type="Hidden" name="UPLOADSIZE_bild1"
        value="50000">
      <input type="File" name="bild1">
    </td>
    <td>
      <span style="color:red">
        <?php
        $objFU = new upload();
        $objFU->copy();
        brecho($objFU->arrResults['UPLOADSIZEERROR_bild1']);
        brecho($objFU->arrResults['UPLOADMIMEERROR_bild1']);
        brecho($objFU->arrResults['UPLOADERROR_bild1']);
        ?>
      </span>
      <span style="color:green">
        <?php
        brecho($objFU->arrResults['UPLOADFULL_bild1']);
        brecho($objFU->arrResults['UPLOADNAME_bild1']);
        brecho($objFU->arrResults['UPLOADSIZE_bild1']);
        brecho($objFU->arrResults['UPLOADMIME_bild1']);
        ?>
      </span>
    </td>
  </tr>
  <tr>
    <td>
      Thmubnail, GIF, maximal 2 KByte<br>
      <input type="Hidden" name="UPLOADDIR_bild2"
        value="user2">
      <input type="Hidden" name="UPLOADMIME_bild2"
        value="image/gif">
      <input type="Hidden" name="UPLOADSIZE_bild2"
        value="2048">
      <input type="File" name="bild2">
    </td>
    <td>
      <span style="color:red">
        <?php

```

```
$objFU = new upload();
$objFU->copy();
brecho($objFU->arrResults['UPLOADSIZEERROR_bild2']);
brecho($objFU->arrResults['UPLOADMIMEERROR_bild2']);
brecho($objFU->arrResults['UPLOADERROR_bild2']);
?>
</span>
<span style="color:green">
<?php
brecho($objFU->arrResults['UPLOADFULL_bild2']);
brecho($objFU->arrResults['UPLOADNAME_bild2']);
brecho($objFU->arrResults['UPLOADSIZE_bild2']);
brecho($objFU->arrResults['UPLOADMIME_bild2']);
?>
</span>
</td>
</tr>
<tr>
<td colspan="2"><input type="Submit" value="Absenden"></td>
</tr>
</table>
</form>
</div>
</body>
</html>
```

Im Formular sind versteckte Felder untergebracht, die als zusätzliche Information übertragen werden. Der Name der Felder bestimmt, wie die Daten interpretiert werden. Dabei wird der Typ der Information als Präfix des Feldnamens übertragen, den Suffix bildet der Name des Dateifeldes. So können beliebig viele Dateien gleichzeitig gesendet und unterschiedlich ausgewertet werden. Folgende Präfixe sind erlaubt:

- UPLOADDIR

Bestimmt ein Unterverzeichnis oder Pfad unterhalb des Installationsverzeichnisses des Skripts, wo die Dateien abgelegt werden.

- UPLOADMIME

Hiermit kann eine Liste von MIME-Typen angegeben werden, die akzeptiert werden.

- UPLOADSIZE

Dieses Feld bestimmt, wie groß die Datei in Bytes maximal sein darf.

Die Klasse *upload*, die im nächsten Listing gezeigt wird, wertet diese Informationen aus und erzeugt entsprechende Fehlermeldungen bzw. gibt die tatsächlichen Informationen über den Erfolg zurück. Die Rückgabe erfolgt in einem in einer Eigenschaft der Klasse gespeicher-

Wie es funktioniert

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

ten Array. Die Indizes des Arrays sind vergleichbar aufgebaut: Der Präfix bestimmt den Datentyp, der Suffix den Dateinamen. Folgende Präfixe sind definiert:

- UPLOADFULL

Hier wird der vollständige (tatsächliche) Pfad gespeichert, unter dem die Datei abgelegt wurde.

- UPLOADNAME

Hier wird der Name der Datei gespeichert.

- UPLOADSIZE

Dieses Element enthält die tatsächliche Größe der Datei.

- UPLOADMIME

Hier kann der MIME-Typ der Datei abgelesen werden, wie er vom System erkannt wurde.

Das nächste Listing zeigt die Definition der Klasse *upload*.

Listing 5.9:
uploadclass:
Universelle Klasse
zum Hochladen von
Dateien

```
<?php
class upload
{
    var $arrPostFiles;
    var $arrPostField;
    var $arrResults;

    function upload()
    {
        $this->arrPostFiles = $GLOBALS['HTTP_POST_FILES'];
        $this->arrPostField = $GLOBALS['HTTP_POST_VARS'];
        $this->arrResults = array();
    }

    function copy()
    {
        if (is_array($this->arrPostFiles))
        {
            foreach($this->arrPostFiles as $strFieldName =>
                                     $arrPostFiles)
            {
                if ($arrPostFiles['size'] > 0)
                {
                    $strFileName      = $arrPostFiles['name'];
                    $intFileSize      = (int) $arrPostFiles['size'];
                    $strFileMIME      = $arrPostFiles['type'];
                    $strFileTemp      = $arrPostFiles['tmp_name'];
                    $strFileTargetDir = $this->arrPostField['UPLOADDIR_' . $strFieldName];
                    $strFileTargetMIME
```

```
        = $this->arrPostField['UPLOADMIME_' . $strFieldName];
        $intFileTargetSize
    = (int) $this->arrPostField['UPLOADSIZE_' . $strFieldName];
    $bInSuccess = TRUE;
    if ($intFileSize > $intFileTargetSize)
    {
        $this->arrResults['UPLOADSIZEERROR_' . $strFieldName]
            = "Upload fehlgeschlagen: Die hochgeladene
            Datei war zu groß
            (<u>$intFileTargetSize</u> Byte
            erlaubt, <u>$intFileSize</u> Byte
            geladen).";
        $bInSuccess = FALSE;
    }
    if (strstr($strFileTargetMIME, $strFileMIME)
        === FALSE)
    {
        $this->arrResults['UPLOADMIMEERROR_' . $strFieldName]
            = "Upload fehlgeschlagen: Dateityp nicht
            akzeptiert (<u>$strFileTargetMIME</u>
            erwartet, <u>$strFileMIME</u>
            gesendet).";
        $bInSuccess = FALSE;
    }
    if ($bInSuccess)
    {
        $this->arrResults['UPLOADFULL_' . $strFieldName]
            = dirname($PATH_TRANSLATED)
            . "$strFileTargetDir/$strFileName";
        $this->arrResults['UPLOADNAME_' . $strFieldName]
            = $strFileName;
        $this->arrResults['UPLOADSIZE_' . $strFieldName]
            = $intFileSize;
        $this->arrResults['UPLOADMIME_' . $strFieldName]
            = $strFileMIME;
        @copy ($strFileTemp, dirname($PATH_TRANSLATED)
            . "$strFileTargetDir/$strFileName");
    } /* end if */
    } else {
        $this->arrResults['UPLOADERERROR_' . $strFieldName]
            = "Upload fehlgeschlagen: Es wurden keine
            Daten übertragen.";
    } /* end if */
    } /* end foreach */
} /* end if */
} /* end function */
} /* end class upload */
?>
```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Wie es funktioniert Die Klasse verwendet den direkten Zugriff auf die globalen Arrays `$HTTP_POST_VARS` (Feldvariablen) und `$HTTP_POST_FILES` (Upload-Informationen), um unabhängig von den Feldnamen arbeiten zu können. Der Zugriff erfolgt einmalig im Constructor der Klasse:

```
$this->arrPostFiles = $GLOBALS['HTTP_POST_FILES'];
$this->arrPostField = $GLOBALS['HTTP_POST_VARS'];
```

Die eigentliche Arbeit erledigt die Methode `copy`. Hier wird zuerst geprüft, ob überhaupt Dateien übertragen werden sollten:

```
if (is_array($this->arrPostFiles))
```

Dann wird das Array durchlaufen, das die Dateiinformationen enthält, sodass alle Dateien erreicht werden.

```
foreach($this->arrPostFiles as $strFieldName =>
    $arrPostFiles)
```

Dann wird die Bearbeitung der Dateien ausgeführt, allerdings nur, wenn der vorab von PHP ausgeführte Upload-Prozess erfolgreich war:

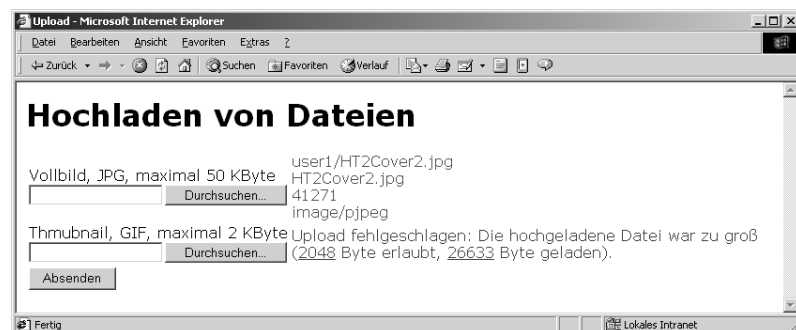
```
if ($arrPostFiles['size'] > 0)
```

Im übrigen Teil der Funktion werden die Vorgaben der versteckten Felder mit den tatsächlichen Werten verglichen und gegebenenfalls entsprechende Fehlermeldungen erzeugt.

Beachten Sie vor dem Test des Skripts, dass die angesprochenen Unterverzeichnisse existieren und der Webserver Schreibrechte hat.



Abbildung 5.3:
Erfolgs- und
Fehlerrückmeldungen des
universellen Uploads



Die Abbildung zeigt, wie im Erfolgsfall die Daten der hochgeladenen Datei und bei Fehlern entsprechende Ausgaben erfolgen.

Weitere Überlegungen

Ergänzt werden könnte die Klasse durch Funktionen, die gleich die passenden Verzeichnisse erstellen. Allerdings können Formulare manipuliert werden – hier wären also zusätzlich noch Sicherheitsmaßnahmen nötig. Außerdem können natürlich weitere Parameter übertragen werden, beispielsweise eine minimale Dateigröße.

5.3 Lösung III: Counter

Counter werden auf vielen Seiten eingesetzt. Oft genügt eine einfache Zählung der Hits. Kommerzielle Counteranbieter haben zwar inzwischen ein ganzes Spektrum an Funktionen um die Counter herum aufgebaut, die schnelle und einfache Zählung von Hits ist jedoch eher eine Nebenfunktion.

5.3.1 Textbasierter Counter

Der hier vorgestellte PHP-Counter bietet die einfachste Form eines zuverlässigen Counters.

Die Speicherung des Zählerstandes erfolgt in einer Textdatei, sodass beim Herunterfahren des Servers kein Datenverlust eintritt. Zum Zurücksetzen des Zählers wird einfach die Datei gelöscht.

Eingesetzte Funktionen für diese Lösung sind:

- Dateioperationen: `file_exists`, `fgets`, `rewind`, `fputs`, `fclose`, `fopen`

`file_exists()`
`fgets()`
`rewind()`
`fputs()`
`fclose()`
`fopen()`

Vorstellung des Codes

Der Counter ist sehr einfach aufgebaut. Das folgende Skript wird in jede Datei eingebunden, deren Aufrufe gezählt werden sollen. Beachten Sie, dass Sie verschiedene Dateinamen (Variable `$counter`) einstellen müssen, wenn Aufrufe getrennt gezählt werden sollen.

```
<?php
function tcounter ()
{
    $counterFile = "access.txt";
    if (file_exists($counterFile))
    {
        $fp = fopen($counterFile, "r+");
        $count = fgets($fp,6);
        $count++;
        rewind($fp);
        fputs($fp,$count,6);
        fclose($fp);
    } else {
        $fp = fopen($counterFile, "w");
        $count = "1";
        fputs($fp,$count,6);
        fclose($fp);
    }
    return $count;
}
?>
```

Listing 5.10:
Einfacher
textbasierter Counter
aus tcounter

Wie es funktioniert Die Vorgehensweise ist ausgesprochen einfach. Zuerst wird die Zählerdatei geöffnet (oder neu angelegt, wenn es sie noch nicht gibt). Dann werden die ersten fünf Zeichen gelesen, als Integer interpretiert, um eins erhöht und wieder an die gleiche Stelle geschrieben.

Diese Variante ist sehr einfach. So könnten Manipulationen am Zähler Laufzeitfehler produzieren. Die Begrenzung auf sechs Stellen dagegen ist leicht zu ändern. Außerdem kann nur ein Zähler in einem Projekt existieren.

Nutzungshinweis Um die Funktion zu nutzen, fügen Sie die Funktion *tcounter.php* in Ihr Projekt ein. Sie können dann an der Stelle, wo der Zählerstand erscheinen soll, die folgende Zeile einsetzen:

```
echo tcounter();
```

Dabei wird der Zählerstand erhöht und angezeigt. Beachten Sie, dass mehrfache Aufrufe den Zähler auch mehrfach erhöhen. Der Zählerstand wird in der Datei »access.text« gespeichert, die in dem Verzeichnis angelegt wird, indem das Skript abläuft. In diesem Verzeichnis müssen Schreibrechte erlaubt sein.



Hinweis

Nach meiner Erfahrung ist die Methode, die Datei neu anzulegen, wenn die Funktion `file_exists` sie nicht erkennt, nicht zuverlässig. Auf einem hochbelasteten Server eines Providers wurde mehrfach die Datei neu angelegt und damit der Zählerstand überschrieben, weil das Dateisystem offensichtlich nicht schnell genug reagierte.

5.3.2 Bildbasierter Counter

Eine andere Variante stellen Counter dar, die Zahlen mit GIF-Bildchen abbilden. Das folgende Skript zeigt nur eine mögliche Implementierung. Die Funktion nutzt die Grafikbibliothek *gd*. Da neuere Bibliotheken GIF nicht mehr unterstützen, werden hier die PNG-Funktionen eingesetzt.

Sie können damit mehrere Counter betreiben. Die Parameter werden in der URL übergeben, wie die folgenden Beispiele zeigen:

```

```

Der folgende Counter bekommt einen roten Hintergrund:

```

```

Im Skript `ICOUNTER.PHP`, das zur Ansteuerung (auf der CD) verwendet wird, ist folgender Aufruf eingesetzt worden:

```

```

Als Zähler wird die bereits vorgestellte Funktion *tcounter* eingesetzt.

Aufbau eines bildbasierten Counters

Das folgende Skript nutzt die image()-Funktionen. Vorzugsweise wird hier das Bildformat PNG genutzt, alternativ kann der Counter auch GIF erzeugen, wenn Ihre GD-Bibliothek dies unterstützt.

```
<?php
header("Content-type: image/png");
require("colors.php");
$BGColor = GetColor($BGColor);
$FGColor = GetColor($FGColor);

// Größe berechnen
$SizeX = $Length * 13;
$SizeY = 16;
// Counter mit führenden Nullen auffüllen
$counter = sprintf( "%0".$Length. "d", $counter);
// Bild erzeugen
$img = ImageCreate($SizeX + 4, $SizeY + 4);
// Transparenz einstellen
$trans = ImageColorAllocate($img, 1, 1, 1);
ImageColorTransparent($img, $trans);
// Hintergrund erzeugen
$col = ImageColorAllocate($img, $BGColor["red"],
$BGColor["green"], $BGColor["blue"]);
ImageFill($img, 1, 1, $col);
// Alle Ziffern ausgeben
$col = ImageColorAllocate($img, $FGColor["red"],
$FGColor["green"], $FGColor["blue"]);
$PosX = 0;
for ($i = 1; $i <= strlen($counter); $i++) {
    ImageString($img, $Font, $PosX + 3, 3,
        substr($counter, $i - 1, 1), $col);
    if ($i != 1) {
        // Trennlinie
        ImageLine($img, $PosX, 0, $PosX, $SizeY + 4, $trans);
    }
    $PosX += 13;
}
// Bild ausgeben (hier: PNG-Format)
Imagepng($img);
ImageDestroy($img);
?>
```

Listing 5.11:
Das Skript zur Er-
zeugung der
Grafiken: imgcounter

Der Code ist schon etwas komplexer als in den vorangegangenen Beispielen. Die wesentlichen Funktionen werden nachfolgend erläutert. Versuchen Sie dennoch, den Code insgesamt zu verstehen.

Zunächst erzeugt das gesamte Skript nur ein einziges Bild. Der Abruf des Bildes führt zugleich zum Zählen. Um dem Browser mitzuteilen, dass ein Bild folgt, wird mit der folgenden Zeile der passende HTTP-Header erzeugt:

Wie es funktioniert

```
header( "Content-type: image/png");
```

Dann werden die Farbcodierungen geladen (auf diese Zuordnung wird hier nicht weiter eingegangen):

```
require( "colors.php");
```

Dann wird der Farbwert zu dem Farbwort ermittelt:

```
$BGColor = GetColor($BGColor);  
$FGColor = GetColor($FGColor);
```

Dann folgt derselbe Ablauf für die Vordergrundfarbe. Die Bildgröße wird aus der Variablen *\$length* berechnet. Interessant ist der Ansatz, die Skalierung fest zu programmieren und die Größe in Schritten von 13 x 16 Pixel zu vergrößern:

```
$SizeX = $Length * 13;  
$SizeY = 16;
```

Der Zählerstand wird in einer Textdatei gespeichert, damit bei Serverausfall oder -stillstand keine Werte verloren gehen. Dies erledigt die Funktion *tcounter()*, die im aufrufenden Skript verwendet wird. Sie übergibt die Variable *\$counter* mit dem Zählerstand.

Zuerst erfolgt eine Formatierung mit führenden Nullen entsprechend der Länge:

```
$counter = sprintf( "%0".$Length."d", $counter);
```

Dann wird das Bild erzeugt:

```
$img = imagecreate($SizeX + 4, $SizeY + 4);  
imageinterlace($img, 1);  
$trans = imagecolorallocate($img, 1, 1, 1);  
imagecolortransparent($img, $trans);  
$col = imagecolorallocate($img, $BGColor["red"],  
                          $BGColor["green"], $BGColor["blue"]);  
imagefill($img, 1, 1, $col);  
$col = imagecolorallocate($img, $FGColor["red"],  
                          $FGColor["green"], $FGColor["blue"]);  
$PosX = 0;  
for ($i = 1; $i <= strlen($Counter); $i++) {  
    imagestring($img, $Font, $PosX + 3, 2 + $i % 2,  
               substr($Counter, $i - 1, 1), $col);  
    if ($i != 1) {  
        // Trennzeichen zeichnen  
        imageline($img, $PosX, 0, $PosX, $SizeY + 4, $trans);  
    }  
    $PosX += 13;  
}
```

Das fertige Bild wird an den Browser gesendet.

```
imagepng($img);  
imagedestroy($img);  
?>
```

Auf der CD bzw. Website finden Sie drei Dateien:



- IMG_COUNTER.PHP

Die bereits vorgestellte Datei zum Erzeugen des Counters.

- ICOUNTER.PHP

Die Datei zur Ansteuerung des Counter (Demo-Datei).

- COLORS.PHP

Die Datei mit der Definition der Farbwerte.

5.4 Lösung IV: E-Mail-Abruf

POP3-Clients abfragen ist ein häufiges Problem. Damit können beispielsweise mehrere E-Mail-Konten zusammengefasst und auf einer Website zum Abruf bereitgestellt werden.

5.4.1 Problemstellung

Wenn Sie viel unterwegs sind, kann der Abruf von E-Mails zum Problem werden. Zwar gibt es viele Anbieter von Free-E-Mail, die auch eine Browserschnittstelle bieten, eine neue E-Mail-Adresse ist aber nicht immer erwünscht. Oft werden eigene Mailserver betrieben, die mit POP3-Clients abgerufen werden.

Der Zugriff auf die E-Mail kann zwar auch per Notebook von unterwegs erfolgen, aber auch das ist nicht immer bei der Hand oder anschließbar.

Der hier vorgestellten Lösung liegt die Idee zugrunde, unterwegs die E-Mail auf dem heimischen Server nur zu lesen und sonst nicht anzufassen. Wenn man wieder zu Hause ist, wird die Mail mit dem normalen POP3-Client gelesen.

Der Vorteil liegt auf der Hand:

- E-Mail wird weiterhin zentral archiviert.
- Der Abruf kann von jedem Internet-Café oder fremden Terminals aus erfolgen, es bleiben keine Daten auf anderen Maschinen.
- Große E-Mails lassen sich vor dem Laden löschen.

Die Realisierung basiert darauf, dass SMTP-Server empfangene E-Mails in irgendwelchen Verzeichnissen ablegen. Im Wesentlichen werden Funktionen genutzt, die Dateien suchen und anzeigen.

Funktionsübersicht

Das Skript nutzt die folgenden Funktionen und Befehle:

- Dateioperationen: fopen, opendir, fclose, file
- Zeichenkettenoperationen: ereg, strpos, substr, strlen
- Formatierungen: quoted_printable_decode
- Steuerbefehle: for, while, break

5.4.2 Code und Erläuterungen

Sie finden zuerst den vollständigen Code und anschließend detaillierte Erläuterungen zu den wichtigsten Abschnitten.

Listing 5.12:
Lesen von E-Mail-
Dateien in der Inbox:
reader

```
<html>
<body>
<h1>E-Mail Reader</h1>
<?php
define("MAILDIR","inbox");
$fp = fopen("login.txt","r");
$login = explode("@",fgets($fp,100));
fclose($fp);
if (((($login[0] == $name) && ($login[1] == $password))
    || ($selectfile<>"")))
{
    $fd = opendir(MAILDIR);
    echo "Dateiliste:\n<br>";
    while ($file = readdir($fd)) {
        if (!preg_match("^\.|",$file)) {
            $nextmail = file(MAILDIR."\".$file);
            $lines = count($nextmail);
            echo "<A HREF=default.php?
                selectfile=\".$file.\">\".$file.\"</A>";
            for ($i=0; $i<=$lines &&
                strlen($nextmail[$i])>2; $i++) {
                if (strpos($nextmail[$i], "ubject:")==1) {
                    echo "<br><b>Betreff:</b>".
                        quoted_printable_decode(
                            substr($nextmail[$i],8));
                }
                if (strpos($nextmail[$i], "rom:")==1) {
                    echo "<br><b>Von:</b>".
                        quoted_printable_decode(
                            substr($nextmail[$i],5));
                }
            }
        }
    }
}
```

```

    }
    echo "<BR>\n";
    if ($file == $selectfile) {
        echo "<div>";
        $headflag = 0;
        for ($i=0; $i<=$lines; $i++) {
            if (strlen($nextmail[$i])==2) {
                $headflag = 1;
            }
            if ($headflag==1) {
                echo quoted_printable_decode(
                    $nextmail[$i]);
                echo "<br>";
                if ($moreline==0 && $i==40) {
                    echo "<BR>
                        <A HREF=default.php
                          ?moreline=1&selectfile=\"$
                          $file.\">
                          Bis Ende anzeigen?</A>";
                    break;
                }
            }
        }
        echo "</div><P>";
    }
    echo "<A HREF=\"\$PHP_SELF?moreline=0&selectfile=$file\">
        Text anzeigen?</A><hr noshade size=1>";
}
}
closedir($fd);
} else {
echo <<<FOOTER
    <H3>Bitte geben Sie Nutzernamen und Kennwort ein:</H3>
    <FORM action="\$PHP_SELF" method="post">
    Nutzernamen: <INPUT type="text" name="name" size=20 maxlength=20>
    <br>
    Kennwort: <INPUT type="password" name="password" size=20>
    <p>
    <INPUT type="Submit" value="Box abfragen">
    </FORM>
FOOTER;
}
</body>
</html>

```

Der erste Schritt besteht in der Definition des Verzeichnisses als Konstante. Ändern Sie diesen Wert, bevor Sie das Skript ausführen.

```

<?php
define("MAILDIR","inbox");

```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Wie es funktioniert

Nun wird die Datei LOGIN.TXT geöffnet, die Nutzernamen und ein Kennwort in der Form *name@kennwort* enthält. Mit Hilfe der Funktion `explode` wird diese Zeichenkette zerlegt.

```
$fp = fopen("login.txt","r");
$login = explode("@",fgets($fp,100));
fclose($fp);
```

Jetzt wird das Kennwort und der Name überprüft, beim ersten Start des Skripts sind die beiden Variablen *\$name* und *\$password* leer, der `if`-Zweig wird nicht ausgeführt. Statt dessen wird das Formular am Ende des Skripts angezeigt. Erst wenn dort der richtige Name und das passende Kennwort eingegeben wurden, wird der Inhalt des Mailverzeichnisses abgerufen.

```
if ((( $login[0] == $name)
    && ( $login[1] == $password))
    || ( $selectfile<>"")) {
```

Als Erstes wird nun ein Handle zum Mailverzeichnis mit `opendir` geöffnet:

```
$fd = opendir(MAILDIR);
echo "Dateiliste:\n<br>"
```

Die `while`-Schleife durchläuft alle Dateien, die in dem Verzeichnis liegen. In jedem Durchlauf wird außerdem eine Datei der Variablen *\$file* zugewiesen (beachten Sie, dass hier `»=«` und nicht `»==«` steht).

```
while ( $file = readdir($fd)) {
```

Der `if`-Befehl blendet die beiden Verzeichniseinträge `».«` und `»..«` aus, die für das aktuelle und das übergeordnete Verzeichnis stehen. Genutzt wird ein einfacher regulärer Ausdruck:

```
if (!ereg("^\.",$file)) {
```

Nun wird mit jedem Durchlauf die komplette Datei in ein Array gelesen (*file*). Außerdem wird die Anzahl der Zeilen festgestellt:

```
$nextmail = file(MAILDIR."\".$file);
$lines = count($nextmail);
```

Zuerst wird der Dateiname als Link ausgegeben. Mit diesem Link kann eine einzelne E-Mail zur Anzeige gebracht werden. Dazu wird der Dateiname in der Variablen *selectfile* übermittelt:

```
echo "<A HREF=default.php?_
selectfile=\".$file.\">\".$file.</A>";
```

Um die Anzeige aussagekräftiger zu machen, wird die E-Mail nach Zeilen durchsucht, die den Absender und Betreff enthalten. Diese Zeilen beginnen mit `»From:«` oder `»Subject:«`. Die Zeilen werden mit einer `for`-Schleife durchlaufen, die als zweites Abbruchkriterium die Zeilenlänge hat. Der Header einer E-Mail ist zu Ende, wenn eine leere

Zeile erscheint und die Länge mit dem Zeilenumbruch kleiner als 2 ist (strlen(\$nextmail[\$i]>2)):

```
for ($i=0; $i<=$lines &&
    strlen($nextmail[$i]>2); $i++) {
```

Die beiden if-Befehle wählen die passenden Zeilen aus:

```
    if (strpos($nextmail[$i], "subject:")==1) {
        echo "<br><b>Betreff:</b>".
            quoted_printable_decode(
                substr($nextmail[$i],8));
    }
    if (strpos($nextmail[$i], "rom:")==1) {
        echo "<br><b>Von:</b>".
            quoted_printable_decode(
                substr($nextmail[$i],5));
    }
}
echo "<BR>\n";
```

Die Anzeige der selektierten E-Mail erfolgt, wenn zuvor auf den oben definierten Link geklickt wurde. In diesem Fall ist die Variable *\$selectfile* mit dem Dateinamen belegt. Damit die Anzeige an der richtigen Stelle erfolgt, wird der Name mit jeder Datei verglichen:

```
if ($file == $selectfile) {
    echo "<div>";
    $headflag = 0;
    for ($i=0; $i<=$lines; $i++) {
        if (strlen($nextmail[$i])==2) {
            $headflag = 1;
        }
        if ($headflag==1) {
            echo quoted_printable_decode($nextmail[$i]);
            echo "<br>";
            if ($moreline==0 && $i==40) {
                echo "<BR>
                <A HREF=default.php?moreline=1&selectfile=\".
                $file.\"> Bis Ende anzeigen?</A>";
                break;
            }
        }
    }
    echo "</div><P>";
}
```

Als letztes folgt das HTML-Formular zur Abfrage von Nutzernamen und Kennwort, was sicher keiner Erläuterung bedarf.

5.5 Lösung V: Skripte für die Website

Gästebücher sind ebenso beliebt wie typisch. Sicher gibt es bessere, fertige Lösungen. Die hier gezeigte Version zeigt die Programmierung mit Textdateien zur Speicherung.

5.5.1 Minigästebuch

Das Minigästebuch ist eine typische »Quick-and-dirty«-Anwendung. Der einfache Aufbau und simple Stil ist hier bewusst gewählt, um zu zeigen, wie man mit wenigen Zeilen eine sinnvolle Funktion programmieren kann. Komplexere Versionen finden Sie in den Datenbankkapiteln.

Code mit Erläuterungen

Im Folgenden finden Sie den Code wieder im Ganzen und anschließend die Erläuterungen der wichtigsten Abschnitte

*Listing 5.13:
guestbook: Ein
einfaches Gästebuch*

```
<h1>Willkommen in unserem Gästebuch</h1>
<h2>Bitte schreiben Sie hier Ihre Nachricht:</h2>
<form action="<?php echo $PHP_SELF ?>" method="POST">
  <textarea cols=40 rows=5 name="note" wrap=virtual></textarea>
  <input type="submit" value=" Nachricht absenden ">
</form>
<?php
if(isset($note)) {
    $fp = fopen("guestbook.txt","a");
    fputs($fp,nl2br($note)."<p>\n");
    fclose($fp);
}
?>
<h2>Was andere bereits hier geschrieben haben:</h2>
<?php @readfile("guestbook.txt") ?>
```

Wie es funktioniert

Die Arbeitsweise ist sehr einfach. Das Skript ruft sich mit dem Absenden des Formulars selbst auf:

```
<form action="<?php echo $PHP_SELF?>" method="POST">
```

Die Systemvariable \$PHP_SELF zeigt immer auf das gerade ausgeführte Skript.

Wenn ein Formularfeld mit POST gesendet wird, speichert PHP den Inhalt automatisch in einer gleichnamigen Variablen. Ob etwas gesendet wurde, erkennt die folgende Zeile:

```
if(isset($note)) {
```

Die Funktion isset prüft, ob eine Variable initialisiert wurde. Ist das der Fall, wird die Datei GUESTBOOK.TXT geöffnet, der Inhalt des Textfeldes hineingeschrieben und die Datei wird wieder geschlossen. Das

gesamte Gästebuch wird, die ältesten Einträge zuerst, mit der folgenden Zeile zur Anzeige gebracht:

```
<?php @readfile("guestbook.txt") ?>
```

In ➡ Abschnitt 8.2 *Gästebuch* ab Seite 609 wird dieses Gästebuch mit Hilfe einer Datenbank etwas erweitert. Die Einträge können dann deutlich flexibler verwaltet werden.

5.5.2 Bilderverwaltung

Die erste Stufe zeigt die Verwaltung von Bilddateien in einem Verzeichnis.

Beschreibung

Die Verwaltung von CD-Covern ist sicher keine typische Webserverlösung. Aber als Bestandteil eines Fanshops oder als erster Schritt zu einer größeren Lösung ist der Ansatz sicher interessant. Wie schon im vorangegangenen Beispiel wurde auf kompakte Programmierung Wert gelegt. Die folgenden Ausbaustufen zeigen, wie leicht Sie Funktionen hinzufügen können.

```
<?php
$fd = opendir("/upload");
while($cover = readdir($fd))
{
    if (preg_match("/\.(jpg|gif|png)$/i", $cover))
    {
        echo "<a href=\"\$cover\">";
        echo "<img src=\"\$cover\" align=middle border=0
            height=80 width=80>";
        echo "</a> $cover<br>\n";
    }
}
closedir($fd);
?>
```

Listing 5.14:
CD-Cover-
Verwaltung:
Bildanzeige: coverpic

Diese erste Variante sucht nur Bilder in einem Verzeichnis, das Sie mit

```
$fd = opendir("/upload");
```

angeben. Mit Hilfe der Schleife wird jedes Bild im Verzeichnis gelesen:

```
while($cover = readdir($fd)) {
```

Der reguläre Ausdruck erkennt an Hand der Dateierweiterung, ob es sich um ein Bild handelt:

```
if (preg_match("/\.(jpg|gif|png)$/i", $entry)) {
```

Die drei echo-Befehle geben dann einen Link auf die Bilder aus und zeigen das passende Bild auch an.

Interessant wäre es nun, wenn die Upload-Funktion genutzt werden könnte, um Bilder auch per Browser hochzuladen.

5.5.3 Bilderalbum mit Upload-Funktionen

Die zweite Stufe erweitert die Bildverwaltung um eine Funktion zum Hochladen von Dateien.

Listing 5.15:
CD-Cover-Ver-
waltung: Upload-
Funktion:
coverpicupload

```
<h1>Willkommen in unserer Bildersammlung</h1>
Senden Sie uns Ihre Bilder!
<p>
<form action="<?php echo $PHP_SELF ?>"
      enctype="multipart/form-data" method=POST>
<input type="hidden" name="MAX_FILE_SIZE" value="500000">
<input type="file" name="file" size=50>
<input type="submit" value="Coverbild senden">
</form>
<?php
$dir = "upload";
if($file)
{
    $dest = "$dir/$file_name";
    if (!copy($file, $dest))
    {
        echo "Fehler - kann Datei nicht ablegen<br>\n";
        exit;
    }
}
$fd = opendir($dir);
while($cover = readdir($fd))
{
    if(preg_match("/\.(jpg|gif|png)$/i", $cover)) {
        echo "<a href=\"$dir/$cover\">";
        echo "<img src=\"$dir/$cover\" align=middle
              border=0 height=80 width=100>";
        echo "</a> $cover<br>\n";
    }
}
closedir($fd);
?>
```

Wie es funktioniert Hier wird die Methode der Auswertung eines Formulars im selben Skript wie beim Gästebuch angewendet. Das `<FORM>`-Tag wurde lediglich um das Attribut

```
enctype="multipart/form-data"
```

erweitert, um auf die Art der übertragenen Daten hinzuweisen. Das versteckte Feld

```
<input type="hidden" name="MAX_FILE_SIZE" value="500000">
```

begrenzt die Dateigröße auf 500 000 Byte. Sie können jeden Wert in Byte angeben. Die Auswahl der Datei von der lokalen Festplatte erfolgt mit

```
<input type=file name=file size=50>
```

Die übertragene Datei wird in einer Variablen abgelegt. Darin unterscheidet sich PHP beispielsweise von ASP, wo übertragene Dateien sofort abgelegt werden. Mit folgender Anweisung wird geprüft, ob wirklich Daten übertragen wurden:

```
if($file) {
```

In der Variablen *\$dest* wird dann der Zielpfad bestimmt:

```
$dest = "$dir/$file_name";
```

Anschließend muss der Inhalt der Variablen *\$file* nur noch kopiert werden:

```
if (!copy($file,$dest))
```

Die if-Abfrage soll lediglich erkennen, ob das Kopieren erfolgreich war. Sie können auch andere Fehlermeldungen einbauen. Denken Sie daran, dass dieser Vorgang für den (anonymen) Webnutzer Schreibrechte im Zielverzeichnis erfordert. Dies ist normalerweise weder bei Unix noch Windows die Standardeinstellung. Die übrigen Zeilen entsprechen dem zuvor bereits gezeigten Code.

Diskussion

Die Sache mit dem anonymen Zugriff wird bei Schreibzugriffen oft nicht erwünscht sein. Sie können das Beispiel leicht um eine Authentifizierung erweitern.

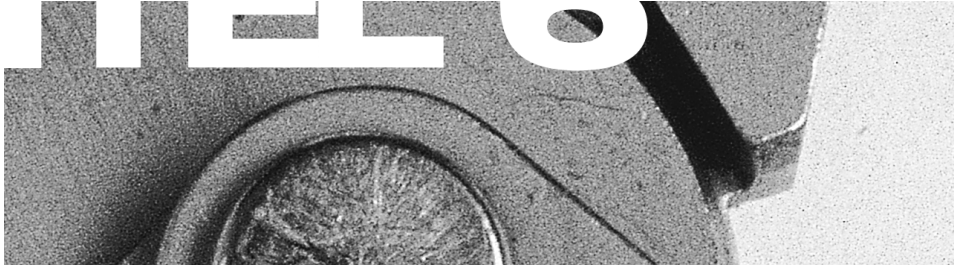
Komfortablere Funktionen zum Hochladen von Dateien finden Sie in ➡ Abschnitt 5.2.3 *Universeller Dateupload* ab Seite 433.

TEIL II



Datenbank- programmierung

6



Grundlagen der Datenbanktechnik

In diesem Kapitel lesen Sie alles über MySQL, das Datenbankmanagementsystem für PHP. Mit MySQL steht ein sehr schnelles, schlankes und leicht handhabbares Programm zur Verfügung. Gezeigt wird der Umgang mit SQL am Beispiel des MySQL-Dialekts und der Kopplung mit PHP. Außerdem wird auf die universelle Schnittstelle ODBC eingegangen, mit der auch auf andere RDBMS zugegriffen werden kann.

Installation von MySQL (Seite 457)

Standardwerkzeuge für Windows (Seite 459)

Der MySQL-Dialekt (Seite 463)

Einführung in SQL (Seite 468)

MySQL-Sicherheit (Seite 514)

ODBC (Seite 518)

6.1 Installation von MySQL

Dieser Abschnitt beschreibt die Einrichtung und Installation des Datenbankmanagementsystems MySQL. Sie können so die Bedingungen auf einem Webserver mit installierter Datenbank auch lokal optimal simulieren.

6.1.1 Installation unter Windows

Die Installation von MySQL ist wenig spektakulär. Sie können zum einen eine Auswahl Binärdistributionen beziehen, die sich einfach installieren lassen. Zum anderen können Sie auch den Quellcode herunterladen und selbst übersetzen. Beiden Varianten liegen Kurzanleitungen bei, die den sehr einfachen Vorgang erklären. Da der häufigere Einsatz der Zugriff auf ein fertig konfiguriertes System beim Provider sein dürfte, soll an dieser Stelle auf eine detaillierte Betrachtung des Feintunings verzichtet werden.

Einrichtung als Dienst unter Windows NT/2000

MySQL kann auch unter Windows NT/2000 optimal betrieben werden. Es ist angebracht, die Software als Dienst zu installieren. So ist MySQL nach dem Start sofort verfügbar. Die Einrichtung ist allerdings nicht vollkommen trivial. Die Installation selbst erfolgt in zwei Schritten:

- Ausführen des Installationsprogramms
- Einrichten des Dienstes
- Anpassen der Konfigurationsdatei

Der letzte Schritt kann entfallen, wenn die Installation in das Standardverzeichnis erfolgt (C:\MYSQL). Schon die Installation auf einer anderen Festplatte verlangt eine Anpassung.

Den Dienst installieren Sie, indem Sie in das Verzeichnis \BIN wechseln und dort folgenden Aufruf starten:

```
C:\Mysql\bin>mysqld --install
```

Zur Konfiguration erstellen Sie eine Datei MY.CNF und legen diese auf dem Laufwerk C: im Stammverzeichnis ab. Zu dieser Position gibt es keine Alternative. Die Datei hat folgenden Aufbau (die Pfade geben die tatsächliche Position des Programms wieder):



Dienst installieren

MySQL konfigurieren

Listing 6.1:
Muster für die Konfigurationsdatei
»my.cnf« unter
Windows

```
[client]
port = 3306

[mysqld]
port = 3306
skip-locking
set-variable = key_buffer=16M
set-variable = max_allowed_packet=1M
set-variable = thread_stack=128K
basedir=D:\\W2K\\Programme\\mysql
datadir=D:\\W2K\\Programme\\mysql\\data
tmpdir=D:\\W2K\\Programme\\mysql\\temp
```

Entscheidend sind nur die drei Parameter basedir, datadir und tmpdir. Achten Sie auf die doppelten Backslashes als Verzeichnistrenner. Eingestellt werden kann hier auch der Port; 3306 ist der Standardport.

Dienst starten



Dienste

Jetzt können Sie den Dienst starten. Außerdem kann der Dienst so eingerichtet werden, dass er beim Hochfahren von Windows automatisch startet. Dazu gehen Sie folgendermaßen vor:

- *Unter NT 4*

SYSTEMSTEUERUNG | DIENSTE. In der Diensteliste wählen Sie MYSQL und dann STARTEN.

- *Unter Windows 2000*

SYSTEMSTEUERUNG | VERWALTUNG | DIENSTE. In der Liste der Dienste wählen Sie MYSQL. Klicken Sie mit der rechten Maustaste und wählen dann aus dem Kontextmenü STARTEN.

Problembehebung

Wenn Sie unter Windows 2000 Startprobleme mit dem Dienst haben, versuchen Sie folgende Lösung:

- Gehen Sie in den EIGENSCHAFTEN-Dialog.
- Wechseln Sie zur Registerkarte ANMELDEN.
- Wählen Sie als Anmeldekonto statt des lokalen Systemkontos den ADMINISTRATOR oder einen Benutzer, der Mitglied in der Gruppe ADMINISTRATOREN ist.

6.1.2 Der MySQL-Monitor

Mit MySQL arbeiten

Wenn Sie das erste Mal mit MySQL arbeiten, sollten Sie nicht sofort die Kombination mit PHP testen, da sich zu viele Fehlerquellen ergeben. Versuchen Sie, erst MySQL alleine zum Laufen zu bringen. Die hier beschriebenen Werkzeuge helfen dabei.

Bedienung des MySQL-Monitors

Mit dem MySQL-Monitor können Sie SQL-Befehle an die Datenbank senden und Ausgaben beobachten. Dieses Werkzeug ist auf allen Plattformen verfügbar und identisch. Es ist zwar vergleichsweise primitiv, andererseits aber schnell und einfach zu bedienen. Sie können den MySQL Monitor mit dem folgenden Befehl starten:

```
mysql
```

Anschließend meldet sich statt des Systemprompts MySQL mit:

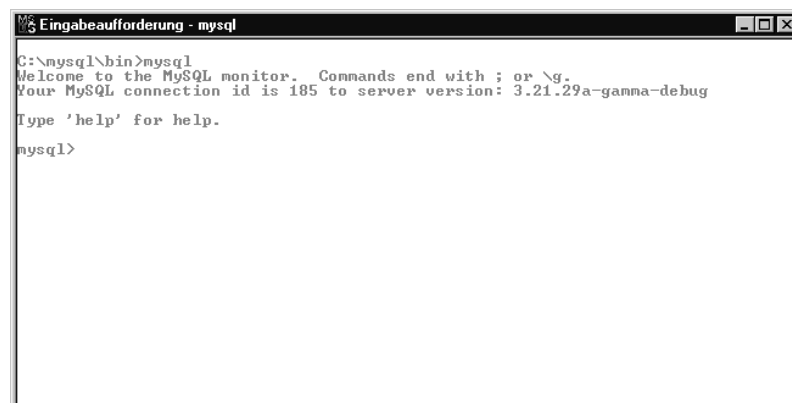
```
mysql>
```

Sie können nun eines oder mehrere Kommandos eingeben. ENTER erzeugt lediglich eine neue Zeile, die mit dem folgenden Symbol gekennzeichnet ist:

```
->
```

Um das Kommando abzuschließen und an die Datenbank zu senden, geben Sie ein Semikolon oder \g ein. Kommandos, die eindeutig enden, werden auch mit ENTER ausgeführt.

Unter Windows starten Sie den MySQL-Monitor in einer DOS-Box. Unter UNIX können Sie MySQL direkt starten.



Probieren Sie zuerst den MySQL-Monitor aus, bevor Sie die Verknüpfung mit PHP verwenden. Fehler in PHP-Skripten können die Reaktion der Datenbank überdecken und erschweren Einsteigern den Umgang mit dem Gesamtsystem.



Hinweis

Abbildung 6.1:
MySQL-Monitor

6.2 Standardwerkzeuge für Windows

Hier werden die mit MySQL gelieferten – also in der Distribution befindlichen – Werkzeuge beschrieben. Informationen zu Werkzeugen unter Linux wurden hier nicht mit aufgenommen. Empfehlenswert ist

dann das Programm PHPMYADMIN, das in ➡ Abschnitt 7.6 *Administration mit phpMyAdmin* ab Seite 586 beschrieben wird.

6.2.1 winMySQLAdmin

Dieses Programm überwacht den MySQL-Dienst und gibt Informationen über die aktuelle Struktur aus. Änderungen an Datenbanken und Tabellen sind nicht möglich.

Aufruf

Das Programm finden Sie in folgendem Pfad:

```
c:\Programme\MySQL\bin\winmysqladmin.exe
```

Der Pfad kann bei der konkreten Installation davon abweichen. Das Programm öffnet sich kurz und verschwindet dann – nicht völlig, sondern als Symbol in den Systemtray der Taskleiste. Klicken Sie mit der rechten Maustaste auf das Symbol und wählen Sie dann im Kontextmenü SHOW ME.

Funktionen

Folgende Funktionen sind verfügbar:

- ENVIRONMENT

Dieser Teil zeigt allgemeine Informationen über den aktuellen Zustand der Umgebung an, beispielsweise Servername, IP-Nummer, Version, Anzahl der Tabellen und Uptime.

- START CHECK

Hier werden die Startbedingungen überprüft und eventuell auftretende Probleme werden angezeigt.

- SERVER

Hier werden die internen Variablen des MySQL-Servers angezeigt.

- MY.INI SETUP

Hier können Sie die Initialisierungsdateien editieren.

- ERR FILE

Dieser Bereich enthält das Fehlerprotokoll.

- VARIABLES

Diese Registerkarte zeigt konfigurierbare Variablen an.

- PROCESS

Hier finden Sie Angaben zu allen derzeit offenen Prozessen.

- DATABASES

Hier können Sie alle Datenbanken und Tabellen mit allen Optionen einsehen.

- REPORT

Hier werden alle Informationen zusammengefasst ausgegeben. Diese Ausgabe kann kopiert und gedruckt werden.

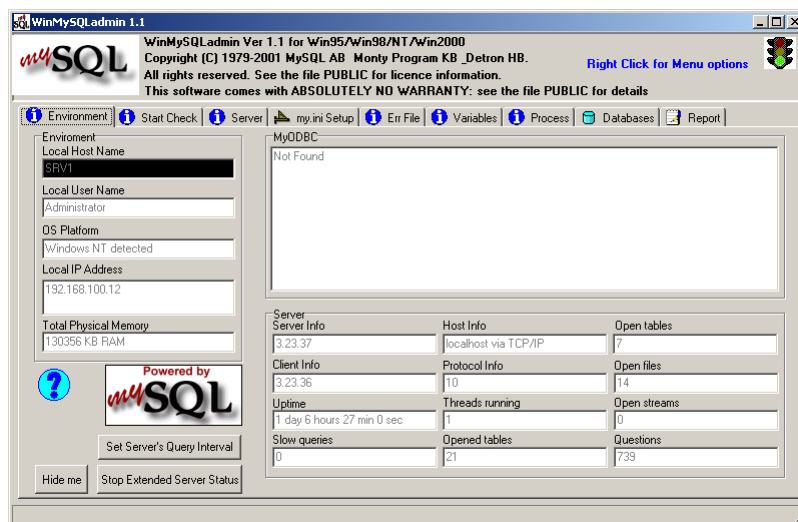


Abbildung 6.2:
winMySQLAdmin

Bis auf die MY.INI-Datei sind Änderungen nicht möglich. Das Programm wird vor allem zur Leistungs- und Verfügbarkeitsüberwachung eingesetzt.

6.2.2 MySQLManager

Der MySQL-Manager erlaubt Änderungen an Tabellen und das Anlegen und Löschen von Datenbanken. Das Werkzeug ist relativ primitiv und kaum für den dauerhaften Einsatz als Entwickler geeignet. Für die ersten Schritte mag es jedoch hilfreich sein.

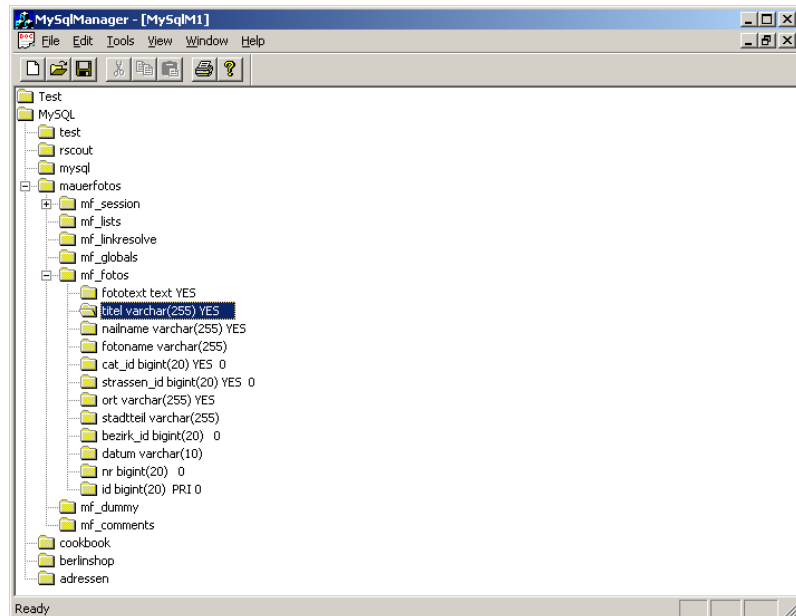
Das Programm finden Sie in folgendem Pfad:

Aufruf

```
c:\Programme\MySQL\bin\mysqlmanager.exe
```

Der Pfad kann bei der konkreten Installation davon abweichen. Das Programm zeigt nach dem Start einen Baum der Datenbankstruktur, der bis herunter zu Spalteninformationen führt.

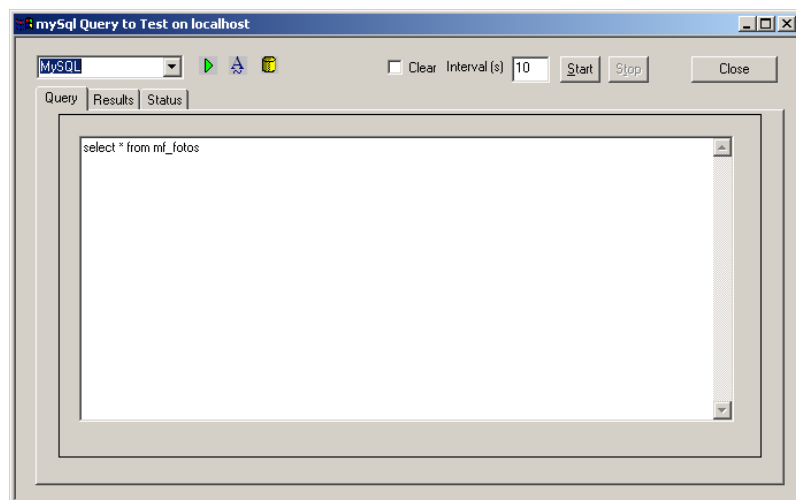
Abbildung 6.3:
MySQLManager



Funktionen

Der MySQLManager bietet als interessante Funktionen die Möglichkeit, SQL-Abfragen direkt auszuführen. Dazu wählen Sie im Menü TOOLS die Option SQL QUERY. Geben Sie die Abfrage im Dialog QUERY ein. Klicken Sie dann auf das grüne Dreieck. Im Bereich RESULT finden Sie dann die Reaktion des MySQL-Servers.

Abbildung 6.4:
SQL Query



Die Schriftart lässt sich mit dem Symbol »A« einstellen.

6.3 Der MySQL-Dialekt

MySQL verfügt über den SQL Standardbefehlssatz. Vielfältige Erweiterungen führen zwar zu einer höheren Leistungsfähigkeit, erfordern aber etwas Lernarbeit, auch wenn Sie SQL bereits kennen.

6.3.1 MySQL versus Standard-SQL

Dieser Abschnitt erklärt einige Besonderheiten von MySQL gegenüber dem SQL Standard ANSI SQL92. Außerdem werden die Einschränkungen gezeigt, die MySQL gegenüber anderen SQL-Servern hat. Diese Abschnitte sind nur von Bedeutung, wenn Sie bereits Erfahrungen im Umgang mit SQL haben und bei der Realisierung an die eine oder andere Funktion denken, MySQL aber die Ausführung verweigert. Anfänger, die sich das erste Mal mit SQL beschäftigen, sollten gleich zum ➡ Abschnitt *Innere Struktur der Datenspeicherung* ab Seite 468 springen. Es werden dort nur die SQL-Befehle gezeigt, die sowohl im ANSI SQL92-Standard als auch von MySQL gesprochen werden. Erweiterungen sind ausdrücklich als solche hervorgehoben. Mit diesem Grundwissen können Sie alle Datenbanken ansprechen. Erweiterungen bestimmter Datenbanken schränken die Portierbarkeit des Codes erheblich ein – doch ist gerade dies einer der Vorteile von PHP.

MySQL entspricht weitestgehend dem Entry Level des Standards **Welche Standards?** ANSI SQL92. Außerdem wird das ODBC Level 0-2 erfüllt.

6.3.2 Erweiterungen gegenüber ANSI SQL92

Die Erweiterungen gegenüber ANSI SQL92 sind nur bei MySQL zu finden. Sie sollten davon nicht unbedingt Gebrauch machen, denn die Portierung der Skripte auf andere Datenbanken wird stark erschwert. Wenn Sie gezielt für MySQL entwickeln und sicher sind, dass ein Wechsel der Datenbank nicht in Frage kommt, können die Erweiterungen unter Umständen elegantere Lösungen ermöglichen.

Es gibt die Möglichkeit, an einigen Stellen MySQL-Erweiterungen zu nutzen und trotzdem portierbaren Code zu entwickeln. SQL kennt Kommentarzeichen der Art `/*! Kommentar */`. Erweiterungen innerhalb dieser Kommentarzeichen werden von allen SQL-Datenbanken ignoriert. MySQL durchsucht aber auch die Kommentare und führt darin enthaltene Anweisungen aus, wenn es sich um eigene Erweiterungen der Sprache handelt. Ein Beispiel dazu:

```
SELECT /*! STRAIGHT_JOIN */ spalte FROM tab1, tab2 ...
```


Liste der Erweiterungen

Die folgende Auflistung nennt alle Erweiterungen:

- Drei Feldtypen sind zusätzlich vorhanden: MEDIUMINT, SET, ENUM.
- Zwei Feldtypen verhalten sich anders: BLOB, TEXT.
- Vergleiche von Zeichenketten unterscheiden standardmäßig nicht zwischen Groß- und Kleinschreibung. Sortierungen basieren auf dem Zeichensatz ISO-8859-1 Latin1. Wenn die Sortierung nach ASCII gewünscht ist, nutzen Sie das Attribut BINARY.
- MySQL bildet jede Datenbank als Verzeichnis und jede Tabelle als Datei ab. Dies führt zu einem plattformabhängigen Verhalten: Auf UNIX-Plattformen wird bei Datei- und Verzeichnisnamen Groß- und Kleinschreibung unterschieden. Dies gilt damit auch für Datenbanknamen und Tabellen. Bei Windows wird dies nicht unterschieden, Entsprechendes gilt für die Benennung von Datenbanken und Tabellen. Damit ist es auch erlaubt, Namen mit Ziffern beginnen zu lassen. Dies erlauben andere Datenbanken nicht.
- LIKE kann auch auf numerische Spalten angewendet werden.
- Der SELECT-Befehl ist um zwei Schlüsselwörter erweitert worden: INTO outfile und STRAIGHT_JOIN.
- Neu sind die Befehle OPTIMIZE TABLE und SHOW.
- GROUP BY muss nicht alle Spalten enthalten, die ausgegeben werden.
- Zusätzlich zu den Operatoren OR und AND können die Symbole || und && genutzt werden. Für die Verknüpfung von Zeichenketten muss anstatt || die Funktion CONCAT verwendet werden.
- Der Operator % kann als Synonym für MOD eingesetzt werden.
- Logische Operatoren können auch auf der linken Seite eines SELECT-Befehls geschrieben werden:

```
SELECT spalte1 >= 100 FROM table
```
- Der Abruf der letzten automatisch generierten ID erfolgt mit der Funktion LAST_INSERT_ID().
- MySQL versteht reguläre Ausdrücke mit REGEXP und RLIKE.
- Diverse neue Funktionen: BIT_COUNT, ELT, FROM_DAYS, FORMAT, IF, PASSWORD, ENCRYPT, PERIOD_ADD, PERIOD_DIFF, TO_DAYS, WEEKDAY.
- Erweiterungen bei GROUP BY-Funktionen: STD, BIT_OR, BIT_AND.
- Neue Befehle: REPLACE ersetzt Kombinationen aus DELETE und INSERT. FLUSH löscht Statusinformationen.

6.3.3 Fehlende Funktionen

MySQL hat einige Einschränkungen, die bei der Portierung von Code, **Version 3.23.xx** der für andere Datenbanken geschrieben wurde, von Bedeutung ist:

- Manche Datenbanken erlauben privaten Speicherplatz für Nutzer, sogenannten Tablespace. Folgende Anweisung ist nicht erlaubt:

```
CREATE TABLE user.mytable ... IN mytablespace
```

- Eingeschlossene (verschachtelte) SELECT-Abfragen sind nicht erlaubt. Die folgende Konstruktion ist in MySQL unzulässig:

```
SELECT * FROM tb1 WHERE id IN (SELECT id FROM tb2)
```

- Anstatt SELECT...INTO TABLE muss das Kommando SELECT...INTO OUTFILE geschrieben werden.
- Transaktionen werden nicht unterstützt. Sie können Tabellen bei kritischen Operationen aber mit LOCK TABLES und UNLOCK TABLES verriegeln. Die Transaktionskommandos BEGIN und COMMIT sind Dummies.
- Gespeicherte Prozeduren werden nicht unterstützt.
- Trigger werden nicht unterstützt.
- Fremdschlüssel (*Foreign Keys*) werden nicht unterstützt. Das Schlüsselwort FOREIGN KEY existiert aus Kompatibilitätsgründen, hat aber keine Funktion.
- Sichten (Views) werden nicht unterstützt.
- Kommentarzeilen mit »--« sind nicht erlaubt.

6.3.4 Ausblick auf MySQL 4

Zum Zeitpunkt der Drucklegung war MySQL 4.0 noch nicht verfügbar. Es gab allerdings bereits eine Liste der »geplanten« Änderungen. Diese wird nachfolgend wiedergegeben. Der Umfang lässt vermuten, dass Programme, die für MySQL 3 geschrieben wurden, unverändert laufen. Die in diesem und dem nächsten Kapitel gezeigten Techniken sind also auch weiterhin gültig. Wenn Ihr Provider auf MySQL 4 umstellt, sollten Sie auf der Homepage von MySQL den konkreten Stand der Änderungen und Erweiterungen abrufen:

```
http://www.mysql.com
```

Geplante Änderungen in MySQL 4.0.0

Geplant ist die Einführung eines neuen Formats für die Daten (.FRM-Dateien). Damit lassen sich später weitere Tabellenoptionen einfügen, ohne das erneute Formatänderungen notwendig sind. Das alte Format

soll aber weiter unterstützt werden, sodass Konvertierungen vorerst nicht notwendig werden. Neue Tabellen werden aber auf jeden Fall im neuen Format erzeugt.

Bessere Bibliothek	Es wird außerdem eine C-Bibliothek des MySQL-Dienstes geben, der die Integration in Anwendungsprogramme erleichtert. Für PHP hat dies vermutlich dahingehend Auswirkungen, dass eine Integration in die Skriptumgebung effektiver ist – allerdings dürfte auch eine erneute Übersetzung bzw. eine neue DLL notwendig werden.
Backup-Funktion	Neu ist eine Backup-Funktion, die nur noch wenig Systemleistung in Anspruch nimmt.
DELETE FROM	DELETE FROM table gibt nun die Anzahl der gelöschten Reihen zurück. Außerdem können mit einem DELETE-Befehl mehrere Tabellen gelöscht werden.
Volltextsuche	Die Volltextsuche ist erweitert und stark beschleunigt worden. Die Funktionen MATCH bzw. AGAINST können nun mit Booleschen Operatoren kombiniert werden, wie sie von Suchmaschinen her bekannt sind (+, -, <, >, ~ und *).
Zeichensätze	Es werden nun mehr Zeichensätze unterstützt, sodass mehrsprachige Angebote leichter verarbeitet werden können. SET CHARACTER_SET transformiert komplette Abfragen, nicht nur einzelner Zeichensätze.
SSL	Verbindungen zu MySQL-Servern können nun auch mit Verschlüsselung arbeiten.
Optimierung	Viele Funktionen sind optimiert worden. Das betrifft beispielsweise Abfragen mit ORDER BY key_name DESC. Strukturabfragen mit SHOW COLUMNS FROM table öffnen nicht mehr die Tabelle, sondern nur noch die Definitionsdateien.

6.3.5 Hinweise für die Portierbarkeit

Einer der großen Vorteile von PHP ist die Portierbarkeit des Codes. Wenn Sie Applikationen schreiben, die MySQL als Datenbank verwenden, sollten Sie die Portierbarkeit beachten. Vor allem der Wechsel von UNIX nach Windows und umgekehrt bereitet immer wieder Probleme. Die folgenden Hinweise helfen Ihnen, solche Probleme zu vermeiden:

- Beachten Sie immer Groß- und Kleinschreibung bei der Benennung von Datenbanken und Tabellen. Es ist eine gute Idee, nur Kleinschreibung zu verwenden.
- Die Speicherung der Datenbanken im Dateisystem legt nahe, Systemfunktionen zur Datensicherung (Backup) zu nutzen. Diese unterscheiden sich jedoch zwischen den Plattformen erheblich. Vermeiden sie daher hardcodierte Backupskripte.

6.3.6 Technische Daten

Die folgende Übersicht zeigt, welche Leistungen MySQL bietet. Sie können damit besser entscheiden, ob sich die Datenbank für den vorgesehenen Einsatzzweck eignet.

- MySQL besitzt Multi-Threading-Unterstützung, profitiert also von Mehrprozessorsystemen.
- Es gibt eine API (Application Programming Interface) für C, C++, Java, Perl, PHP, Python und TCL.
- MySQL unterstützt viele Plattformen, unter anderem Linux, Solaris, Windows.
- Es gibt sehr viele Datentypen: Ganzzahlen mit oder ohne Vorzeichen und 1, 2, 3, 4 oder 8 Bytes Länge, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET und ENUM Typen.
- JOINS sind sehr schnell. Alle SQL-Funktionen sind geschwindigkeitsoptimiert.
- Unterstützt werden auch GROUP BY und ORDER BY, COUNT(), AVG(), STD(), SUM(), MAX() und MIN().
- ODBC (Open DataBase Connectivity) steht für Windows9x/NT und 2000 zur Verfügung. Damit kann auch mit VBScript oder Visual Studio auf MySQL zugegriffen werden, was die Entwicklung erheblich vereinfacht.
- Es gibt bis zu 16 Indizes pro Tabelle. Jeder Index darf aus 1 bis 15 Spalten der Teilsalten bestehen. Die maximale Indexlänge beträgt 256 Bytes.
- Datensätze fester und variabler Länge werden unterstützt.
- Große Datenbanken werden unterstützt, es gibt Musteranwendungen mit 50 000 000 Datensätzen und mehr.
- Alle Spalten haben Standardwerte (bei INSERT).
- MySQL wurde in C und C++ geschrieben und ist auf vielen Compilern kompilierbar.
- Aliase auf Tabellen und Spalten entsprechen dem SQL92-Standard.
- DELETE, INSERT, REPLACE und UPDATE geben zurück, wie viele Datensätze bearbeitet wurden.
- Nutzer können sich per TCP/IP, Unix-Sockets oder Named Pipes unter NT/2000 mit MySQL verbinden. Sind Webserver und

MySQL-Server getrennt, genügt eine Internetverbindung zwischen beiden.

- MySQL ist Jahr-2000-fähig.

Innere Struktur der Datenspeicherung

Die kompletten Daten werden vom Betriebssystem in Dateien abgelegt. Standardmäßig werden drei Dateierweiterungen verwendet:

- FRM

Diese Datei enthält die Definition der Tabelle (Form).

- ISD

Hier werden die eigentlichen Daten gespeichert.

- ISM

Diese Datei enthält den Index der Tabelle.

Standardmäßig finden Sie die angelegten Datenbanken in jeweils einem eigenen Verzeichnis unterhalb von /DATA. Der Name des Verzeichnisses entspricht dem Namen der Datenbank.

6.4 Einführung in SQL

SQL – *Structured Query Language* – ist die wichtigste Datenbankabfragesprache. Trotz vielfältiger Erweiterungen gibt es einen gut standardisierten Sprachkern, der von allen anderen Datenbanksystemen verstanden wird. Es lohnt sich als Webprogrammierer unbedingt, diesen Kernwortschatz SQL zu erlernen.

6.4.1 Grundsätzliche Konzepte

Bevor Sie mit SQL arbeiten, müssen Sie die grundlegenden Konzepte verstanden haben, die hinter Datenbanken stecken. Erst dann wird es gelingen, erfolgreich eigene Projekte zu entwickeln.

Datenbanken und deren Bestandteile



Wenn Sie mit Datenbanken umgehen, werden Ihnen immer wieder dieselben Begriffe begegnen. In der Literatur und auch unter Fachleuten herrscht nicht immer Einigkeit, mit welchem Begriff welche Eigenschaft benannt wird. Zum einen liegt dies in der verwirrenden Vielfalt, zum anderen an der unscharfen Definition. Die folgende Erläuterung stellt die Grundlage für dieses Buch dar.

Eine Datenbank ist eine Sammlung von Datenbankobjekten, die neben den eigentlichen Daten auch weitere Hilfsinformationen und Anweisungen enthalten kann. Wesentlicher Bestandteil einer Datenbank sind die Tabellen, in denen der Betreiber die Daten ablegt. In der Datenbank werden aber auch Prozeduren, Zugriffsrechte und Verknüpfungen hinterlegt. Man spricht deshalb von Datenbankobjekten.

Woraus besteht eine Datenbank?

Tabellen sind Datenbankobjekte, in denen die Daten abgelegt sind. Tabellen werden durch eine Anordnung von Spalten und Reihen dargestellt. Wenn eine komplette Reihe aus der Tabelle extrahiert wird, um die darin enthaltenen Daten zu betrachten oder zu manipulieren, wird von einem Datensatz gesprochen. Wenn wir aus dem Datensatz eine bestimmte Spalte auswählen, nennt man dies ein Feld. Daraus resultieren oft Missverständnisse. Felder können, müssen aber nicht gleich Spalten sein. Und Datensätze können, müssen aber nicht gleich Reihen sein. Die Unterscheidung ist dann von Bedeutung, wenn davon gesprochen wird, ob Befehle »spaltenweise« oder »reihenweise« arbeiten. Ebenso können Datentypen nur für eine Spalte und nicht für ein Feld eingestellt werden. Abbildung 6.5 erläutert die Zusammenhänge noch einmal.

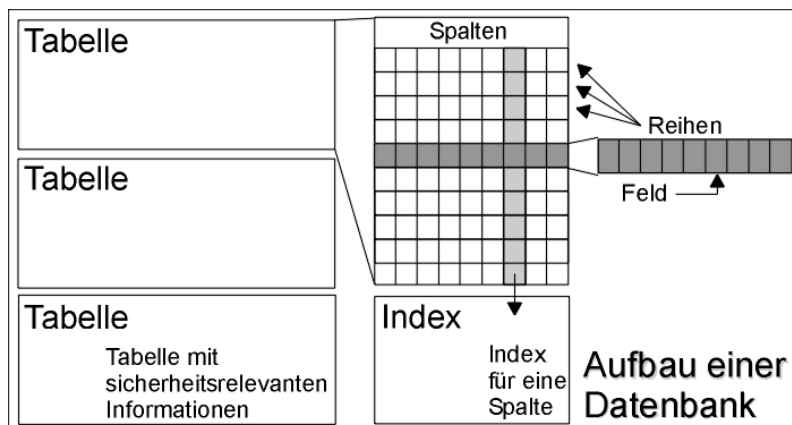


Abbildung 6.5:
Aufbau einer
Datenbank und
Benennung der
Tabellenelemente

In MySQL können Sie – volle Zugriffsrechte vorausgesetzt – beliebig viele Datenbanken anlegen. Wenn Sie MySQL bei einem Provider verwenden, steht Ihnen möglicherweise nur eine Datenbank zur Verfügung. Sie können dann aber beliebig viele Tabellen anlegen, sodass dies in der Praxis nur selten eine problematische Einschränkung darstellt.

Relationale Datenbanken und SQL

Praktisch sind SQL-Abfragen kleine Befehlszeilen oder Befehle, die an den Server gesendet werden. Die Befehle sind in englischer Sprache und sollten den Sinn der Abfrage erkennen lassen. Sie können in SQL auch kleine Programme schreiben. Allerdings ist SQL ganz streng auf

SQL

die Bedienung von Datenbanken ausgerichtet. SQL alleine macht kaum Sinn. Sinnvoll ist die Integration in eine Programmier- oder Skriptumgebung, beispielsweise PHP.

MySQL kann natürlich schon als vollwertige Datenbank verstanden werden. Sie können Tabellen anlegen und mit dem interaktiven Werkzeug MySQL-Monitor Befehle und Kommandos eintippen. Für die praktische Arbeit ist das natürlich kaum sinnvoll. Für Test- und Übungszwecke ist es dagegen eine wertvolle Einrichtung, denn Sie können die Fehlerquellen zwischen Datenbank und Nutzerschnittstelle ausschließen.

RDBMS

MySQL gehört zur Klasse der RDBMS (relational database management system). Diese Datenbanken ordnen Daten in Tabellen an und verwalten zwischen den Tabellen bestehende Beziehungen (engl. relations). Durch solche Beziehungen – wir werden sie später als Schlüssel bezeichnen – sind komplexe Abfragen mit vergleichsweise einfachen Befehlen möglich.

Eine Datenbank ist eine Sammlung verschiedener Objekte, davon sind die Tabellen mit den eigentlichen Daten nur ein Teil. Die folgenden Begriffe sollten Ihnen im Laufe der Zeit vertraut werden, wenn Sie mit MySQL arbeiten:

Wichtige Begriffe

- Datenbanken (*databases*)
Dies sind Verzeichnisse, in denen die Datenbankobjekte liegen.
- Tabellen (*tables*)
Diese Objekte enthalten die Daten.
- Standardwerte (*defaults*)
Dies sind Standardwerte, die eingesetzt werden, wenn bei der Eingabe eines neuen Datensatzes ein Feld nicht besetzt wird.
- Regeln (*rules*)
Mit Regeln werden zulässige Inhalte für Felder beschrieben.
- Indexe (*indexes*)
Mit einem Index wird der Datenzugriff beschleunigt und Such- oder Sortierfunktionen werden vereinfacht.
- Datentypen (*data types*)
Jede Spalte einer Tabelle kann nur Daten in einem ganz bestimmten Format aufnehmen, dem Datentyp.

- Beschränkungen oder Limitierungen (*constraints*)

Damit werden die zulässigen Wertebereiche der Daten eingeschränkt und unlogische Eingaben vermieden.

Um die einzelnen Befehle zu verstehen, mit denen Datenbanken erzeugt und gesteuert werden, sind einige Blicke auf die grundlegenden Konzepte notwendig.

Das relationale Modell und die Datennormalisierung

Das relationale Datenbankmodell verlangt eine ganz bestimmte Form von internen Beziehungen (Relationen) zwischen den Daten. Bei der Wahl der Tabellenstruktur werden Sie damit konfrontiert, Ihre Daten in Tabellen zu speichern. Denken Sie dabei an ein einfaches Bestellsystem – sicher eine sehr typische Anwendung. Die folgende Tabelle zeigt eine Reihe eindeutiger Bestellvorgänge, die einige Personen ausgelöst haben. Die Tabelle enthält alle Informationen, die für die Abwicklung benötigt werden (Namen, Anschriften, Bestelldaten):

id	Vorname	Nachname	Nr.	Zeile	Artikelnr.	Preis
101	Clemens	Krause	100	1	335548	100,00
102	Clemens	Krause	100	2	665882	50,00
103	Clemens	Krause	100	3	775511	99,90
104	Janine	Bünning	101	1	445566	44,00
105	Caroline	Schröder	102	1	335548	100,00
106	Caroline	Schröder	102	2	665882	50,00
107	Caroline	Schröder	102	3	335548	100,00
108	Janine	Bünning	103	1	775511	99,00

Normalisierung von Daten

Abbildung 6.6:
Eine nicht
normalisierte Tabelle

Die Tabelle enthält sowohl die Namen als auch die Daten der Artikel und die Bestellinformationen. Stellen Sie sich die nicht gezeigten Spalten für die Adresse und die Artikelbeschreibung usw. rechts weiterführend vor. Diese Tabelle ist offensichtlich sehr unübersichtlich und – was noch schwerer wiegt, – sie enthält redundante Daten. So tauchen sowohl die Namen als auch Artikelnummern und Preise mehrfach auf.

Um die Tabelle in eine für relationale Datenbanken passende Form zu bringen, werden die Daten *normalisiert*. Dazu werden folgende fünf Regeln angewandt:

Normalisierungsregeln

- *Eliminierung sich wiederholender Gruppen*

Für jede Gruppe von Daten, die sich wiederholen, legen Sie eine eigene Tabelle an. Im Beispiel haben Sie drei Gruppen: *Adressen*, *Bestellungen* und *Artikel*. Also werden drei Tabellen angelegt. Jede

Tabelle bekommt einen eindeutigen Schlüssel, das heißt, eine eindeutige Spalte.

- *Eliminierung redundanter Daten*

Wenn eine Eigenschaft mehrere Werte annehmen kann, bringen Sie diese in eine eigene Tabelle. Wenn Sie im Beispiel feststellen, dass jeder Artikel mehrere Preise haben kann, legen Sie zusätzlich eine Tabelle *Preise* an.

- *Eliminieren Sie Spalten, die von keinem Schlüssel abhängen*

Wenn Eigenschaften keinen Zusammenhang mit dem Schlüsselfeld haben, also nicht ebenso eindeutig zugeordnet werden können, übertragen Sie diese Eigenschaften in eine eigene Tabelle. Wenn Sie im Beispiel zwei Adressen pro Kunden haben (Lieferanschrift und Rechnungsanschrift), können Sie die Adresse nicht mehr eindeutig einem Schlüsselfeld *Kunden-ID* zuordnen. Trennen Sie die Anschriften des Kunden in weitere Tabellen ab.

- *Isolieren Sie unabhängige Beziehungen*

Keine Tabelle darf zwei oder mehr Beziehungen haben, die nicht direkt abhängig sind.

Die oben gezeigte Tabelle sollten Sie also in mindestens drei Tabellen splitten:

Abbildung 6.7:
Tabelle Adressen

id	Vorname	Nachname
101	Clemens	Krause
102	Janine	Bünning
103	Caroline	Schröder

Abbildung 6.8:
Tabelle Artikel

id	Artikelnr.	Preis
101	335548	100,00
102	665882	50,00
103	775511	99,90
104	445566	44,00

Abbildung 6.9:
Tabelle Bestellungen

id	Kundennr.	Bestellnr.	Zeile	Artikelnr.
101	101	100	1	335548
102	101	100	2	665882
103	101	100	3	775511
104	102	101	1	445566

id	Kundennr.	Bestellnr.	Zeile	Artikelnr.
105	103	102	1	335548
106	103	102	2	665882
107	103	102	3	335548
108	102	103	1	775511

Zwischen den so geschaffenen Tabellen definieren Sie Beziehungen. Abbildung 6.10 zeigt die möglichen Verknüpfungen. Genutzt werden hier sogenannte 1-n-Beziehungen, das heißt, auf der einen Seite sind die Schlüssel eindeutig, auf der anderen nicht. Es gibt auch 1-1- und n-n-Beziehungen.

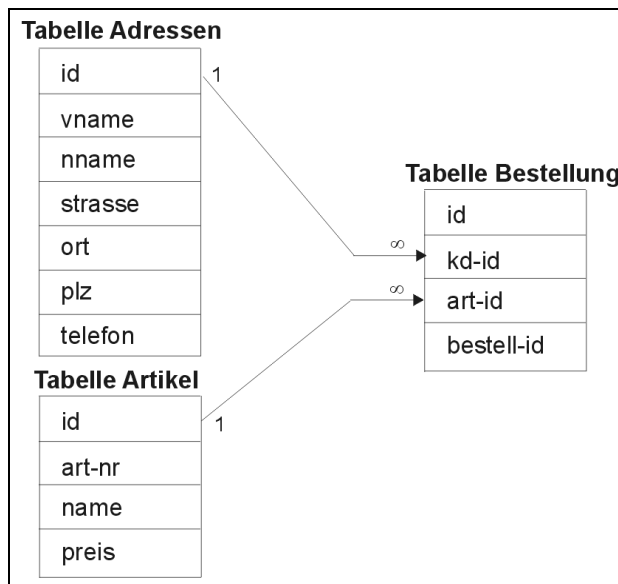


Abbildung 6.10:
Beziehungen
zwischen Tabellen
durch Schlüsselfelder

Große Tabellen, wie die am Anfang gezeigte nicht normalisierte Tabelle, kosten Systemleistung. Die Aufteilung in sehr viele kleine Tabellen (extreme Normalisierung) kompliziert aber auch die Abfragen, was ebenso die Systemleistung reduziert. Versuchen Sie, einen Mittelweg zu finden (wie im Beispiel, wo die Bestelltabelle, die seltener angesprochen wird, nicht weiter normalisiert wurde).



Hinweis

Beziehungen

Nichts geht im Leben ohne Beziehungen, bei SQL-Datenbanken auch nicht. Sie haben bereits die Begriffe 1-1-Beziehung, 1-n-Beziehung und n-n-Beziehung kurz gesehen. Die vorgestellte Mustertabelle enthält eine 1-n-Beziehung. Dabei sind jedem Schlüssel der linken Tabelle ein oder mehrere Datensätze in der rechten Tabelle zugeordnet. Der Kun-

de selbst ist eindeutig, aber er kann natürlich beliebig viele Bestellungen aufgeben.

n-1-Beziehung

Bei der n-1-Beziehung kann der Schlüssel auf der linken Seite mehrfach auftreten. Stellen Sie sich vor, bei der Eingabe der Adressen wird das Bundesland verlangt. Es wäre überaus hilfreich, wenn dies bei der Eingabe aus einer Liste ausgewählt werden könnte. Speichern Sie die Bundesländer in einer Tabelle, entsteht eine n-1-Beziehung, denn Sie können ein Bundesland bei mehreren Adressen auswählen (jede Adresse enthält ein Bundesland, und dies wiederholt sich irgendwann). In der Tabelle der Bundesländer selbst ist jeder Eintrag dagegen eindeutig und einmalig.

1-1-Beziehung

Eine 1-1-Beziehung entsteht, wenn Sie zu jedem Kunden weitere Attribute (beispielsweise das Geburtsdatum) speichern, die nur sehr selten benötigt werden. Es macht dann Sinn, diese Angaben in einer separaten Tabelle unterzubringen. Zwischen der Tabelle *Adressen* und der Tabelle *Daten* besteht dann eine 1-1-Beziehung, denn in beiden Tabellen ist jeder Datensatz eindeutig zugeordnet.

n-n-Beziehung

Eine n-n-Beziehung könnte bei einem anderen Fall entstehen. Wenn Sie die Bestellungen betrachten, so treten die Bestellnummern mehrfach auf (n-mal). Das liegt an der bewusst nicht perfekten Normalisierung. Jede Gruppe von Bestellnummern wird aber immer zusammen versendet, kann also einer weiteren Tabelle mit Versandarten zugeordnet werden. Auch diese Liste ist nicht eindeutig, wenn Sie zu jedem Versand individuelle Daten erfassen, beispielsweise persönliche Lieferwünsche.

6.4.2 Theorie der SQL-Sprache



SQL – die Structured Query Language – kann in drei Kategorien unterteilt werden, die in der Literatur auch als selbstständige »Sprachen« betrachtet werden. Diese Sprachen sind alle Bestandteil der SQL:

DDL
DML
DCL

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Data Control Language (DCL)

Die Data Definition Language (DDL) ist die Sprache mit Befehlen zur Erzeugung und Manipulation von Datenbankobjekten. Darunter werden Befehle verstanden, mit denen Sie Datenbanken, Tabellen, Sichten oder Prozeduren anlegen und bearbeiten können. Beispiele sind die Befehle `CREATE DATABASE` oder `USE`.

Die Data Manipulation Language (DML) umfasst folgende Befehle:

- INSERT

Dieser Befehl fügt einer Tabelle neue Datensätze hinzu.

- UPDATE

Dieser Befehl ändert bestehende Daten in einer oder mehreren Spalten und in einem oder mehreren Datensätzen.

- DELETE

Mit diesem Befehl löschen Sie Daten aus einer Tabelle.

- SELECT

Dieser Befehl wählt Daten entsprechend der Anforderung aus.

Weniger umfangreich ist die Data Control Language (DCL), zu der Befehle zur Transaktionssteuerung und Benutzerverwaltung gehören.

6.4.3 Literale

Wenn Sie Befehle für MySQL schreiben, egal ob in PHP oder direkt am MySQL-Prompt, sind bestimmte Schreibweisen zu beachten. Dieser Abschnitt erklärt die wichtigsten Fälle.

Schreibweisen

Zeichenketten werden in MySQL entweder durch einfache oder doppelte Anführungszeichen gekennzeichnet: **Zeichenketten**

```
'Dies ist eine Zeichenkette'  
"Dies ist eine Zeichenkette"
```

Innerhalb dieser Zeichenketten gibt es einige Sonderzeichen, so genannte Escape-Zeichen. Diese werden mit dem Backslash eingeleitet. MySQL erkennt die folgenden Escape-Zeichen: **Sonderzeichen**

- \0

Entspricht dem ASCII-Wert 0 (NUL).

- \n

Zeilenumbruch (newline) ASCII CHR(13).

- \t

Tabulator, ASCII CHR(9).

- \r

Wagenrücklauf ASCII CHR(10).

- \b
Rückschritt.
- \'
Einfaches Anführungszeichen.
- \"
Doppeltes Anführungszeichen.
- \\
Das Backslash-Zeichen.
- \%
Das Prozent-Zeichen (% ist in SQL ein Platzhaltersymbol).
- _
Der Unterstrich (_ ist in SQL ein Platzhaltersymbol).

Um Anführungszeichen in Zeichenketten verwenden zu können, gibt es mehrere Wege:

- Das ' schreiben Sie doppelt: ''
- Das " schreiben Sie doppelt: ""
- Setzen Sie vor das Anführungszeichen das Escape-Zeichen: \"
- Setzen Sie jeweils das entgegengesetzte Anführungszeichen ein, benötigen Sie keine besondere Behandlung:

```
"Jetzt schlägt's 13"
```

```
'Dies ist ein 19"-Monitor'
```

Wenn Sie Binärdaten direkt in ein BLOB-Feld schreiben, müssen Sie die folgenden Zeichen als Escape-Sequenz (mit \ vorangestellt) angeben:

- NUL (ASCII 0)
- \ (ASCII 92)
- ' (ASCII 39)
- " (ASCII 34)

Zahlen

Zahlen werden mit Dezimalpunkt geschrieben, wenn es sich um Fließkommazahlen handelt. Ganzzahlen (Integer) dürfen keinen Punkt enthalten.

Gültige Ganzzahlen sind:

```
9284
0
-78
```

Gültige Fließkommazahlen sind:

```
45.98
-4567e+5
2.0
```

Das spezielle Schlüsselwort `NULL` repräsentiert nicht 0, sondern steht für undefinierte Felder oder nicht vorhandene Daten. Die leere Zeichenkette `""` ist nicht `NULL`! Wenn Sie Textfelder exportieren und `NULL` definiert ausgeben müssen, verwenden Sie den Code `\N`.

Nullwerte

Namenskonventionen

Namen für Datenbanken, Tabellen, Indizes, Spalten und Aliase müssen folgenden Regeln genügen:

- Besteht aus alphanumerischen Zeichen, dem Unterstrich und dem `$`-Zeichen. Der Standardzeichensatz ist ISO-8859-1 Latin1.
- Datenbanken, Tabellen, Indizes und Spalten dürfen 64 Zeichen lang sein, Aliase dürfen 256 Zeichen lang sein.
- Namen dürfen mit Ziffern beginnen, müssen aber wenigstens einen Buchstaben enthalten. Aus Kompatibilitätsgründen sollten Ziffern nicht am Anfang verwendet werden. Ebenso sind Kombinationen mit dem Fließkommaoperator `e` zu vermeiden (beispielsweise `1e`).
- Punkte sind in Namen nicht erlaubt.

Spalten können in drei Varianten referenziert werden:

Schreibweisen für Spalten

- `spalte`
- `tabelle.spalte`
- `datenbank.tabelle.spalte`

Die erweiterte Referenzierung ist nur nötig, wenn die Namensauswahl nicht eindeutig ist, weil beispielsweise zwei angesprochene Tabellen Spalten gleichen Namens enthalten.

In MySQL spielt Groß- und Kleinschreibung keine Rolle. Da die Datenbanken und Tabellen jedoch im Dateisystem abgelegt werden, ist die Schreibweise entscheidend, wenn das Betriebssystem dies verlangt. Das ist bei allen UNIX-Versionen der Fall. Achten Sie deshalb immer auf Groß- und Kleinschreibung. Windows-Nutzer sollten dies auch beachten, um portierbaren Code zu schreiben.

Groß- und Kleinschreibung

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

6.4.4 Datentypen

Bei der Einführung in PHP waren Datentypen kein echtes Thema, denn PHP kennt Datentypen nur intern. SQL-Datenbanken gehen mit Datentypen sehr viel strenger um. Zwar werden auch hier Datentypen intern umgewandelt, prinzipiell muss aber jede Spalte exakt benannt werden. Entsprechend groß ist die Auswahl an Datentypen.

Regeln für die Auswahl

Sie sollten den Datentyp immer so streng wie möglich festlegen. Wenn Sie Werte im Bereich 0 bis 12 speichern, bietet sich der Typ TINYINT an, der mit der Speichergröße 1 Byte Werte von -128 bis +127 speichert. Genauso wäre auch INT oder FLOAT geeignet, TINYINT stellt aber die engste Annäherung an den Wertebereich dar. Die strengere Auslegung offenbart Programmierfehler eher und zwingt zu sauberer Programmierung. Die Übersicht verwendet folgende Symbole:

- M kennzeichnet die maximale Anzahl angezeigter Stellen. Der höchstmögliche Wert ist 255.
- D steht bei Fließkommazahlen für die Anzahl der angezeigten Dezimalstellen (Stellen nach dem Komma).
- Z steht für das Wort ZEROFILL.
- UNSIGNED sorgt dafür, dass kein Vorzeichen verwendet wird.

MySQL only

Angaben, die in eckigen Klammern [] stehen, sind optional. Die Angabe der Datentypen erfolgt beim Erzeugen von Tabellen. Wenn Sie in der Marginalspalte den Hinweis »MySQL only!« finden, ist der Datentyp eine spezielle Erweiterung in MySQL, wird also von anderen RDBMS wahrscheinlich nicht unterstützt.

- TINYINT [(M)] [UNSIGNED] [Z]
Kleine Ganzzahlen, entweder von 0 bis 255 oder von -128 bis +127.
- SMALLINT [(M)] [UNSIGNED] [Z]
Ganzzahlen, entweder von 0 bis 65 535 oder von -32 768 bis +32 767.
- MEDIUMINT [(M)] [UNSIGNED] [Z]
Ganzzahlen, entweder von 0 bis 16 777 215 oder von -8 388 608 bis +8.388.607.
- INT [(M)] [UNSIGNED] [Z]
Ganzzahlen, entweder von 0 bis 4 294 967 295 oder von -2 147 283 648 bis +2 147 283 647. INTEGER ist ein Alias für INT.

MySQL only!

- `BIGINT [(M)] [UNSIGNED] [Z]`

Ganzzahlen, entweder von 0 bis 18 446 744 073 709 551 615 oder von -9 223 372 036 854 775 808 bis +9 223 372 036 854 775 807.

- `FLOAT(precision) [Z]`

Fließkommazahl, immer vorzeichenbehaftet. *precision* kann 4 oder 8 sein; 4 steht für einfache Genauigkeit, 8 für doppelte. `FLOAT(4)` entspricht `FLOAT`, `FLOAT(8)` entspricht `DOUBLE` (siehe dort).

- `FLOAT[(M,D)] [Z]`

Der Wertebereich reicht von -3,402823466³⁸ bis -1,175494351⁻³⁸, die 0 und +1,175494351⁻³⁸ bis +3,402823466³⁸.

- `DOUBLE[(M,D)] [Z]`

Der Wertebereich umfasst die Zahlen von -1,7976931348623157³⁰⁸ bis -2,2250738585072014⁻³⁰⁸, die 0 und von -2,2250738585072014⁻³⁰⁸ bis +1,7976931348623157³⁰⁸. `DOUBLEPRECISION[(M,D)] [Z]` und `REAL[(M,D)] [Z]` sind Aliase für `DOUBLE`.

- `DECIMAL(M,D) [Z]`

Ungepackte Fließkommazahl, immer vorzeichenbehaftet. Zahlen werden, analog `CHAR`, als Zeichenkette gespeichert, jede Ziffer steht in einem Byte. Das Komma, Vorzeichen usw. belegen jeweils ein Byte. Anwendbar ist dies für die Rechnung mit exakten Werten, bei Währungsangaben oder Messwerten. `NUMERIC` ist ein Alias für `DECIMAL`.

- `DATE`

Datum im Format »YYYY-MM-DD«. Der Wertebereich geht von 1.1.1000 bis zum 31.12.9999.

- `DATETIME`

Datum und Zeit im Format »YYYY-MM-DD hh:mm:ss«.

- `TIMESTAMP[(M)]`

Zeitstempel, Wertebereich von 1.1.1970 bis zum 31.12.2036. Für die Angabe des Anzeigebereiches gilt:

- 14: `YYYYMMDDhhmmss`
- 12: `YYMMDDhhmmss`
- 8: `YYYYMMDD`
- 6: `YYMMDD`

	<ul style="list-style-type: none"> • TIME Zeit im Wertebereich von -838:59:59 bis +838:59:59. Ausgabeformat hh:mm:ss. • YEAR Jahr, Wertebereich 1901 bis 2155, Ausgabe YYYY. • CHAR(M) [BINARY] Zeichenkette fester Länge M. Wertebereich 1 bis 255. Leerzeichen am Ende werden automatisch für die Ausgabe entfernt. Sortieren und Selektieren berücksichtigt Groß- und Kleinschreibung nicht, wenn Sie nicht BINARY verwenden. • VARCHAR(M) [BINARY] Zeichenkette variable Länge, maximal M. Wertebereich 1 bis 255. Leerzeichen am Ende werden automatisch für die Ausgabe entfernt. Sortieren und Selektieren berücksichtigt Groß- und Kleinschreibung nicht, wenn Sie nicht BINARY verwenden.
MySQL only!	<ul style="list-style-type: none"> • TINYBLOB, TINYTEXT BLOB oder TEXT mit maximal 255 Byte. • BLOB, TEXT BLOB oder TEXT mit maximal 65 535 Byte.
MySQL only!	<ul style="list-style-type: none"> • MEDIUMBLOB, MEDIUMTEXT BLOB oder TEXT mit maximal 16 777 215 Byte (16 MByte).
MySQL only!	<ul style="list-style-type: none"> • LONGBLOB, MEDIUMTEXT BLOB oder TEXT mit maximal 4 294 967 295 Byte (4 GByte).
MySQL only!	<ul style="list-style-type: none"> • ENUM('wert1', 'wert2', 'wert3',...) Eine Aufzählung. Ein Feld dieses Typs kann nur eine Zeichenkette enthalten, die einem Objekt der Aufzählung entspricht.
MySQL only!	<ul style="list-style-type: none"> • SET('wert1', 'wert2', 'wert3',...) Wie ENUM, kann aber mehrere Werte aus der Liste enthalten.

Speicherbedarf

Datenbanken können sehr groß werden. Auch wenn heute der Preis einer Festplatte untergeordnet ist, macht eine Berechnung manchmal dennoch Sinn. Denn große Tabellen und große Datenbanken bringen Performanceprobleme mit sich. Damit Sie berechnen können, was Sie erwartet, nutzen Sie die folgenden Tabellen.

Datentyp	Speicherbedarf in Byte
TINYINT	1
SMALLINT	2
MEDIUMINT	3
INT, INTEGER	4
BIGINT	8
FLOAT(4), FLOAT	4
FLOAT(8), DOUBLE, REAL	8
DECIMAL(M,D), NUMERIC(M,D)	M, wenn M>D, sonst D+2

Tabelle 6.1:
Speicherbedarf
numerischer
Datentypen

Datentyp	Speicherbedarf in Byte
DATETIME	8
DATE	3
TIMESTAMP	4
TIME	3
YEAR	1

Tabelle 6.2:
Speicherbedarf Zeit-
und Datumstypen

Datentyp	Speicherbedarf in Byte
CHAR(M)	M
VARCHAR(M)	L + 1, L ≤ M
TINYBLOB, TINYINT	L + 1, L ≤ 2 ⁸
BLOB, INT	L + 1, L ≤ 2 ¹⁶
MEDIUMBLOB, MEDIUMINT	L + 1, L ≤ 2 ²⁴
LOB, LONGINT	L + 1, L ≤ 2 ³²
ENUM	1 oder 2
SET	1, 2, 3, 4 oder 8

Tabelle 6.3:
Speicherbedarf der
Zeichenkettentypen
(L ist der tatsächliche
Inhalt)

Optionen numerischer Datentypen

Bei der Vorstellung der numerischen Datentypen wurden zwei Attribute aufgeführt: UNSIGNED und ZEROFILL. Mit dem Attribut UNSIGNED werden vorzeichenlose Zahlen eingeführt, der absolute Wertebereich vergrößert sich entsprechend. ZEROFILL betrifft dagegen nur die Ausgabe von Werten und füllt die maximal erlaubte angezeigte Stellenzahl nach Links mit Nullen auf. Wenn sie

**UNSIGNED
ZEROFILL**

INT(5)

definieren und den Wert 4 in einem Feld speichern, werden bei der Ausgabe so lange Nullen aufgefüllt, bis 5 Stellen erreicht sind: 00004.

Die Angabe der Ausgabebreite macht nur Sinn, wenn darauf mit ZEROFILL Bezug genommen wird. Generell werden die Werte bis zur erlaubten Feldgröße gespeichert, auch wenn die Ausgabebreite überschritten wird. Beispiel: INT(3) speichert auch den Wert 5324.

Optionen der Zeichenkettentypen

BINARY

Das Attribut BINARY wirkt sich nur auf die Sortierung aus, mit der Angabe wird binär sortiert, also nach dem ASCII-Wert der Zeichen. Andernfalls wird alphabetisch sortiert. Die Angabe wirkt auch auf logische Ausdrücke, die einzelne Zeichen auswerten.

Die Datentypen ENUM und SET

Die Datentypen ENUM und SET sind eine spezielle Erweiterung in MySQL. Sie sollten nur eingesetzt werden, wenn eine Datenbank-unabhängige Programmierung nicht notwendig ist.

ENUM

ENUM ist ein Aufzählungstyp. Bei der Definition können bis zu 65 535 Werte angegeben werden. Sie können in diesem Feld nur eine Auswahl aus diesen Werten speichern. Beispiel:

```
ENUM ("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun")
```

Dieser Aufzählungstyp kann nur die Kurzform von Wochentagen speichern. NULL ist ein zulässiger Aufzählungswert. Wenn Sie NULL angeben, kann es auch gespeichert werden. Wenn NULL nicht erlaubt ist, wird der erste Wert der Liste zum Standardwert.

SET

Auch SET ist ein Aufzählungstyp. Im Gegensatz zu ENUM können aber mehrere Werte aus der Werteliste gespeichert werden. Wenn die Definition zusammen mit NOT NULL erfolgt, wird mindestens eine leere Zeichenkette als Wert eingetragen. Intern werden mehrere Werte als Zeichenketten gespeichert, die durch Kommata getrennt sind. Beispiel:

```
SET ("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun")
```

Diese Definition erlaubt Angaben der Form ("Sat", "Sun").

6.4.5 Datenbanken und Tabellen erzeugen

Bevor mit einer Datenbank gearbeitet werden kann, muss sie erzeugt werden. SQL besteht aus einfachen Befehlen, welche die auszuführende Aufgabe gut reflektieren. Oft erschließt sich der Sinn durch den Befehl selbst.



Tip

Die folgenden Beispiele können Sie nachvollziehen, indem Sie direkt mit MySQL arbeiten. Dies ist einfacher als über die PHP-Schnittstelle.

Die hier besprochenen Techniken sind aber die Grundlage für die Kombination PHP + MySQL, die in Kapitel 7 besprochen wird.

Die Darstellung der Sprache SQL wurde so weit gekürzt, wie für ein globales Verständnis möglich ist. Ausführlichere Informationen finden Sie im MySQL-Handbuch, das auf der CD zu finden ist. Empfehlenswert sind auch Bücher speziell über SQL.



Hinweis

Anlegen der Datenbank: CREATE DATABASE

Der erste Schritt besteht im Anlegen der Datenbank. Dazu verwenden Sie den Befehl `CREATE DATABASE`. Der einzige Parameter ist der Name der Datenbank:

**CREATE
DATABASE**

```
CREATE DATABASE [IF NOT EXISTS] db_name
```

Syntax

Im Datenverzeichnis wurde ein neues Verzeichnis angelegt, das den Namen der Datenbank trägt.

Anlegen einer Tabelle: CREATE TABLE

Der nächste Schritt besteht darin, Leben in die Datenbank zu bringen. Dazu werden Tabellen angelegt. Der Befehl hat folgende Syntax:

CREATE TABLE

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [(definition,...)]
    [table_options] [select_statement]
```

Syntax

Jeder *definition*-Eintrag definiert eine Spalte der Tabelle nach folgender Syntax:

```
col_name type [NOT NULL | NULL] [DEFAULT default_value]
    [AUTO_INCREMENT]
    [PRIMARY KEY] [reference_definition]
| PRIMARY KEY (index_col_name,...)
| KEY [index name] (index_col_name,...)
| INDEX [index name] (index_col_name,...)
| UNIQUE [INDEX] [index name] (index_col_name,...)
| FULLTEXT [INDEX] [index name] (index_col_name,...)
| [CONSTRAINT symbol] FOREIGN KEY index_name (index_col_name,...)
    [reference_definition]
| CHECK (expr)
```

Mit *type* wird der Datentyp der erzeugten Spalte bestimmt, dafür sind folgende Angaben zulässig:

```
TINYINT[(length)] [UNSIGNED] [ZEROFILL]
| SMALLINT[(length)] [UNSIGNED] [ZEROFILL]
| MEDIUMINT[(length)] [UNSIGNED] [ZEROFILL]
| INT[(length)] [UNSIGNED] [ZEROFILL]
| INTEGER[(length)] [UNSIGNED] [ZEROFILL]
| BIGINT[(length)] [UNSIGNED] [ZEROFILL]
| REAL[(length,decimals)] [UNSIGNED] [ZEROFILL]
```

```

| DOUBLE[(length,decimals)] [UNSIGNED] [ZEROFILL]
| FLOAT[(length,decimals)] [UNSIGNED] [ZEROFILL]
| DECIMAL(length,decimals) [UNSIGNED] [ZEROFILL]
| NUMERIC(length,decimals) [UNSIGNED] [ZEROFILL]
| CHAR(length) [BINARY]
| VARCHAR(length) [BINARY]
| DATE
| TIME
| TIMESTAMP
| DATETIME
| TINYBLOB
| BLOB
| MEDIUMBLOB
| LONGBLOB
| TINYTEXT
| TEXT
| MEDIUMTEXT
| LONGTEXT
| ENUM(value1,value2,value3,...)
| SET(value1,value2,value3,...)

```

Mehr Optionen

Dies ist der Teil des Syntaxdiagramms, mit dem man in der Praxis häufiger konfrontiert wird. Mehr Auskunft gibt das Handbuch zu MySQL.

Für das Beispiel wird eine Tabelle erzeugt, die Adressen aufnehmen kann:

```

CREATE TABLE address
(id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
sname VARCHAR(50) NOT NULL,
fname VARCHAR(50) NOT NULL,
street VARCHAR(50),
zip CHAR(5),
city VARCHAR(30),
email VARCHAR(100),
birthday DATE)

```

Bevor Sie den Befehl ausführen, machen Sie die soeben angelegte Datenbank zur aktuellen:

```
USE phonebook
```

Geben Sie nun den Befehl zum Anlegen der Tabelle ein. MySQL gibt folgende Bestätigung aus:

```

mysql> CREATE TABLE address
-> <id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
-> sname VARCHAR(50) NOT NULL,
-> fname VARCHAR(50) NOT NULL,
-> street VARCHAR(50),
-> zip CHAR(5),
-> city VARCHAR(30),
-> email VARCHAR(100),
-> birthday DATE>
-> ;
Query OK, 0 rows affected (0.06 sec)

```

Die Ausschrift »0 rows affected« ist korrekt, denn Reihen (*rows*) wurden bei diesem Befehl noch nicht erzeugt oder bearbeitet.

Um die Einträge verwalten zu können, wurde das Attribut `AUTO_INCREMENT` eingesetzt. Sie müssen sich über die eindeutige Zuordnung nun keine Gedanken mehr machen. Wie am Anfang der SQL-Einführung bereits angesprochen, kennt SQL keine Zeilennummern. Um Datensätze unterscheiden zu können, auch wenn die verwendeten Spalten inhaltlich übereinstimmen, werden oft IDs eingesetzt. Damit Sie nicht im Skript umständlich die nächste freie ID ermitteln müssen, können Sie eine Spalte als `AUTO_INCREMENT` definieren. MySQL erzeugt dann innerhalb des Wertebereiches (im Beispiel `INT`) automatisch die nächste freie Nummer. Die Anwendung ist nur bei Datentypen zulässig, die eindeutig hochgezählt werden können (alle Ganzzahltypen).

AUTO_INCREMENT

Eine Spalte einer Tabelle kann einen Primärschlüssel enthalten. Dies ist das primäre Sortiermerkmal. Wenn `AUTO_INCREMENT` eingesetzt wird, muss diese Spalte den Primärschlüssel `PRIMARY KEY` enthalten.

PRIMARY KEY

Die beiden Attribute `NULL` und `NOT NULL` bezeichnen, ob `NULL` als Wert erlaubt ist oder nicht. Wenn Sie später Daten in die Tabelle einfügen, wird MySQL den Befehl zurückweisen, wenn mit `NOT NULL` gekennzeichnete Spalten keine Werte erhalten. Kennzeichnen Sie nur die Spalten mit `NOT NULL`, die unbedingt etwas enthalten müssen. `NULL` ist optional und kann weggelassen werden, da dies der Standardwert ist.

**NULL
NOT NULL**

Datenbanken und Tabellen löschen: DROP

Manchmal ist es notwendig, Datenbanken oder Tabellen zu löschen. Damit kann der Ursprungszustand in einem System wieder hergestellt werden. Tabellen werden oft auch nur temporär angelegt, um Daten bequem zwischenspeichern zu können.

Mit dem Befehl `DROP TABLE` löschen Sie die angegebene Tabelle:

DROP TABLE

```
DROP TABLE [IF EXISTS] table1 [, table 2, ...]
```

Syntax

Sie können mehrere Tabellen in einem Schritt löschen. Das optionale Attribut `IF EXISTS` unterdrückt Fehlermeldungen, wenn die Tabelle nicht existiert.

Mit dem Befehl `DROP DATABASE` löschen Sie die angegebene Datenbank:

DROP DATABASE

```
DROP DATABASE [IF EXISTS] database
```

Syntax

Das optionale Attribut `IF EXISTS` unterdrückt Fehlermeldungen, wenn die Datenbank nicht existiert.

Als Ausgabe erhalten Sie die Anzahl der gelöschten *Dateien*. Da jede Tabelle drei Dateien umfasst, werden bei 10 Tabellen 30 Dateien gelöscht.

Tabellen ändern: ALTER TABLE

ALTER TABLE

Der Weg »Löschen und Neu-Anlegen« ist wenig praktikabel, wenn die Tabelle bereits Daten enthält. Der Befehl `ALTER TABLE` kann eine Tabelle grundlegend ändern. Intern wird eine neue Tabelle angelegt, die Daten werden aus der alten Tabelle kopiert, dann wird die alte Tabelle gelöscht und die neue Tabelle in den alten Namen umbenannt. Die Parameter entsprechen den bei `CREATE TABLE` verwendeten.

Zusätzlich sind folgende Attribute möglich (das Zeichen `|` steht für oder; Sie können immer nur eines der Attribute anwenden):

Syntax

```
ALTER [IGNORE] TABLE tablename
  ADD [COLUMN] create_definition [FIRST | AFTER col]
| ADD INDEX index_name (column, column,...)
| ADD PRIMARY KEY (column)
| ADD UNIQUE index_name (column, column,...)
| ALTER [COLUMN] column {SET DEFAULT | DROP DEFAULT}
| CHANGE [COLUMN] old_name create_definition
| MODIFY [COLUMN] create_definition
| DROP [COLUMN] column
| DROP PRIMARY KEY
| DROP INDEX name
| RENAME AS new_table_name
| ORDER BY column
| table_options
```

Dabei haben die Attribute folgende Bedeutung:

- `ADD` fügt das entsprechende Element der Tabelle hinzu.
- `ALTER` setzt den Standardwert neu.
- `CHANGE` ändert eine Spaltendefinition und benennt die Spalte um.
- `MODIFY` ändert eine Spaltendefinition (ohne Umbenennung).
- `DROP` löscht das Element.
- `RENAME` benennt die Tabelle um.

Es ist in MySQL möglich, mehrere Attribute der Art `ADD`, `ALTER`, `DROP` und `CHANGE` zu kombinieren. Die Liste wird durch Kommata getrennt. `IGNORE` ist eine MySQL-Erweiterung und unterdrückt den Abbruch des Kommandos, wenn Spalten mit dem Attribut `UNIQUE` (eindeutige Werte) durch das Einfügen von Daten dem nicht mehr entsprechen. Statt des Abbruchs werden die Reihen, die nicht einzuordnen sind, ignoriert (nicht kopiert).

Das Schlüsselwort `COLUMN` dient nur der besseren Lesbarkeit und kann entfallen. Die Erweiterung `FIRST`, `AFTER col` bezeichnet die Position, an der die neue Spalte eingefügt wird.

Im Beispiel wäre eine Erweiterung um Telefonnummern sinnvoll. Geben Sie dazu den folgenden Befehl ein:

```
mysql> ALTER TABLE address
-> ADD COLUMN phone VARCHAR(100)
-> ;
Query OK, 1 row affected (0.17 sec)
Records: 1 Duplicates: 0 Warnings: 0
```

Datenbank verwenden: USE

Vor der ersten Verwendung muss eine Datenbank ausgewählt werden. Dazu wird der Befehl USE eingesetzt:

```
USE database
```

Syntax

6.4.6 Mit den Daten arbeiten

Das Anlegen der Tabellen und Eingeben der Daten wird in der Praxis ein eher einmaliger Vorgang sein. Viel häufiger werden Tabellen abgefragt oder Spalten manipuliert. Entsprechend umfangreicher fallen die Möglichkeiten der Sprache SQL aus.

SELECT

SELECT ist der komplexeste Befehl in SQL und der am häufigsten verwendete. Er tritt jedoch auch in sehr einfacher Form auf. Lassen Sie sich von dem folgenden Syntaxdiagramm nicht erschrecken:

```
SELECT [STRAIGHT JOIN] [SQL_SMALL_RESULT] [SQL_BIG_RESULT]
[SQL_BUFFER_RESULT] [HIGH_PRIORITY]
[DISTINCT | DISTINCTROW | ALL]
select_expression,...
[INTO {OUTFILE | DUMPFILE} 'file_name' export_options]
[FROM table_references
  [WHERE where_definition]
  [GROUP BY {unsigned_integer | col_name | formula}
    [ASC | DESC], ...]
  [HAVING where_definition]
  [ORDER BY {unsigned_integer | col_name | formula}
    [ASC | DESC],...]
  [LIMIT [offset,] rows]
  [PROCEDURE procedure_name]
  [FOR UPDATE | LOCK IN SHARE MODE]]
```

Syntax

Die einfachste Abfrage ist die Ausgabe aller Datensätze einer Tabelle; mit dem folgenden Befehl werden alle Telefonnummern aus der Tabelle *phone* angezeigt:

```
SELECT * FROM phone
```

Diese Form ist die häufigste aller Abfragen. Da SELECT nicht durch besondere Einfachheit glänzt, ist es am Anfang oft einfacher, alle Fel-

SELECT * FROM

der abzurufen und dann in PHP zu selektieren. Profis verwenden natürlich komplexe Variationen des SELECT-Kommandos, denn es ist ausgesprochen leistungsstark. Für die Praxis gibt es signifikante Vorteile, solche Abfragen in die Datenbank auszulagern und nicht »extern« zu erledigen.

Sie können auch mehrere Tabellen abfragen. Wir haben zwei Tabellen und möchten alle Namen der Kunden und die Telefonnummern wissen:

```
SELECT sname, fname, number FROM address, phone
```

Dies bringt nicht unbedingt das gewünschte Ergebnis. Lesen Sie in den folgenden Abschnitten, welche Auswahlmöglichkeiten es gibt. Das Beispiel diente nur der Darstellung der Syntax zur Auswahl mehrerer Spalten und Tabellen.

SELECT * FROM ... WHERE

Datenbankabfragen oder, allgemein, Abfragen (*query*), kennen Sie sicher schon. Jede Anfrage an die Suchmaschinen AltaVista oder Yahoo löst eine Datenbankabfrage aus. Viele dieser Abfragefelder kennen Boolesche (logische) Operatoren, UND (engl. and), ODER (engl. or), NICHT (engl. not) usw. So können Sie in AltaVista nach »'PHP' AND 'MySQL'« suchen. Nur jene Datensätze werden ausgegeben, die »PHP« UND »MySQL« im Suchtext hatten (beachten Sie die einfachen Anführungszeichen zum Kombinieren der Worte zu einer Phrase). In SQL schreiben Sie die Abfrage, bezogen auf unsere Mustertabellen, folgendermaßen:

```
SELECT * FROM address WHERE fname='Krause'
```

Der Befehl besteht aus zwei Schlüsselworten. Das Schlüsselwort SELECT (deutsch Auswahl) leitet den Befehl ein, FROM (deutsch aus) wählt die angesprochene Tabelle (wir haben noch eine zweite Tabelle!) und WHERE (deutsch wo, wobei) ist die Bedingung (Abfragen nach Datensatznummern gibt es ja nicht!). Hinter WHERE können Sie logische Bedingungen formulieren, ganz so wie mit dem IF-Befehl aus vielen Programmiersprachen. Variablen gibt es (zumindest am Anfang) nicht, die hier benutzt werden, sondern nur Spaltennamen. Jeder Spaltenname kann verwendet werden.

Wenn Sie nur ganz bestimmte Spalten benötigen, aber alle Datensätze sehen möchten, verwenden Sie die folgende Schreibweise:

```
SELECT id, sname FROM address WHERE fname='Krause'
```

SELECT ... WHERE

Im ersten Beispiel wurden die beiden Tabellen einfach gemischt; eine Zuordnung zwischen den Tabellen *address* und *phone* ist nicht zu erkennen, wenn Sie die gedachte Verknüpfung zwischen den »ID«-Spalten nicht explizit angeben. Das folgende Beispiel ordnet die Telefonnummern den richtigen Namen zu:

```
SELECT id, sname FROM address, phone WHERE address.id = phone.id
```

So einfach ist das! All diese Beispiele wählen nach Reihen, also Datensätzen, aus. Es gibt aber auch Befehle, die spaltenweise arbeiten.

Einfügen von Daten: INSERT

Die SQL-Befehle zum Füllen mit Werten sind ein recht umständlicher Weg, Daten einzugeben. Auch hier gilt jedoch, dass Sie im Umgang mit diesen Befehlen sicher sein müssen, um den Einsatz in PHP-Skripten zu erwägen.

Nicht immer können Sie auf Tabellen zurückgreifen, die bereits Daten enthalten. Mit INSERT fügen Sie selbst Daten hinzu. Der Befehl hat folgenden Aufbau:

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
         [INTO] tbl_name [(col_name,...)]
         VALUES (expression,...),(...),...
```

Syntax

Die Daten können nicht nur der direkten Angabe, sondern auch einer andern SQL-Abfrage mit SELECT entstammen:

```
INSERT [LOW_PRIORITY] [IGNORE] [INTO] tbl_name
         SELECT select_definition
```

Syntax

Im ➡ Abschnitt zu SELECT (Seite 487) finden Sie die Syntax dieses Befehls. Eine dritte Möglichkeit ist der Einbau von Ausdrücken, was die Aufbereitung von Daten für die Abspeicherung erleichtert:

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
         [INTO] tbl_name
         SET col_name=expression [, col_name=expression, ...]
```

Syntax

Die Spaltennamen müssen nur angegeben werden, wenn *nicht alle* Spalten angesprochen werden. Ohne Angabe geht MySQL davon aus, dass Sie alle Spalten einfügen möchten. Im nächsten Beispiel können Sie dies nicht verwenden, denn dann müssten Sie auch einen Wert in die Spalte *id* einfügen und dies ist wegen des AUTO_INCREMENT-Attributs nicht zulässig.

Die Reihenfolge der Namen spielt keine Rolle, muss also nicht der Definition mit CREATE entsprechen. Die Werte im VALUE-Teil müssen dagegen exakt der Reihenfolge der Spaltenliste folgen.

Geben Sie den folgenden Befehl ein, um einige Adressen in der Datenbank zu haben:

```
mysql> INSERT INTO address
->   (sname, fname, street, zip, city, email, birthday)
->   VALUES
->   ("Jörg", "Krause", "Planufer 93", "10967", "Berlin",
->   "joerg@krause.net", 1964-26-05)
-> ;
Query OK, 1 row affected (0.03 sec)
```

Die Syntax des Befehls INSERT ist einfach. Beachten Sie jedoch, dass die Anzahl der Spalten mit den Werten übereinstimmen muss. Ebenso ist die Reihenfolge entscheidend. Zeichenketten müssen in Anführungszeichen stehen.

Anwendung bei fehlenden Spalten

Die Anwendung ist unproblematisch. Lediglich bei der Behandlung fehlender Spalten und der Einbindung von Standardwerten sind einige mögliche Konstellationen zu beachten:

- Ist ein Standardwert definiert, und Sie schreiben einen eigenen Wert hinein, wird der Standardwert überschrieben.
- Wird eine Spalte mit Standardwert nicht beschrieben, wird der Standardwert eingetragen.
- Wird eine Spalte ohne Standardwert nicht beschrieben, trägt MySQL dort NULL ein. Ist NULL aber nicht erlaubt, wird der Befehl nicht ausgeführt und ein SQL-Fehler erzeugt.
- Hat die Spalte die Eigenschaft `auto_increment`, wird der Wert dennoch übernommen. Dabei sind wieder mehrere Fälle zu unterscheiden:
 - Ist der Schlüssel bereits vorhanden, meldet MySQL einen Fehler »Duplicate entry for key ...«.
 - Ist der Schlüssel größer als alle vorhandenen, wird der nächste von `auto_increment` vergebene Wert diesem größten + 1 entsprechen; die Reihe der Werte enthält dann Lücken.
 - Fehlende Werte lassen sich jederzeit auffüllen, `auto_increment` wird dennoch mit dem größten Wert der Liste + 1 fortfahren.

Löschen von Daten: DELETE

DELETE

Neben dem Einfügen von Datensätzen ist auch das gezielte Löschen möglich. Dazu wird der Befehl DELETE eingesetzt:

Syntax

```
DELETE [LOW_PRIORITY] FROM tbl_name  
[WHERE where_definition]  
[LIMIT rows]
```

Lassen Sie die Bedingung weg, werden alle Datensätze gelöscht. Die Selektion der zu löschenden Datensätze erfolgt mit der WHERE-Bedingung, die ebenso wie bei SELECT beschrieben eingesetzt wird. Sie können deshalb mit SELECT zuvor prüfen, welche Datensätze zurückgegeben werden. DELETE wird dann genau diese löschen, wenn dieselbe Bedingung verwendet wird. Die Angabe der Bedingung ist optional – ohne Einschränkung wird die Tabelle vollständig geleert.

Ändern von Daten: UPDATE

Neben dem Einfügen und Löschen können Sie Datensätze auch ändern. Dazu wird UPDATE eingesetzt. Sie müssen die zu ändernden Spalten angeben und den neuen Wert. Der neue Wert kann auch durch eine Formel definiert werden. Welche Datensätze geändert werden, entscheidet wieder die WHERE-Bedingung:

UPDATE

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name
  SET col_name1=expr1 [, col_name2=expr2, ...]
  [WHERE where_definition]
  [ORDER BY ...]
  [LIMIT rows]
```

Syntax

Hinter SET können Sie mehrere Spalten ansprechen, indem der gesamte Ausdruck durch Kommata getrennt wird:

```
UPDATE phone
  SET name = 'Krause', tel = '123454'
  WHERE id = 1324
```

Wenn Sie die WHERE-Bedingung weglassen, wird die Operation auf alle Reihen der Tabelle angewendet.

6.4.7 Auswahlbedingungen für Abfragen

Sehen Sie sich den folgenden Zugriff auf die Tabelle *products* an. Sie enthält einige Artikel:

```
SELECT name, price FROM products
```

Sie können sich nun alle Preise mit und ohne Mehrwertsteuer ansehen:

```
SELECT price AS netto, price*1.16 AS brutto FROM products
```

Bislang erhalten Sie immer alle Datensätze der Tabelle – mit oder ohne berechnete Spalte. Interessanter ist jedoch die Auswahl bestimmter Datensätze oder die Anzeige in einer definierten Reihenfolge. Die entsprechenden Schlüsselwörter werden nachfolgend vorgestellt.

Sortieren; ORDER BY ... DESC | ASC

Für eine bestmögliche Übersicht können Sie die Ausgabe sortieren lassen:

```
SELECT name, price
  FROM products
 ORDER BY price DESC
```

Mit DESC (von *descend*) wird absteigend sortiert und mit ASC (von *ascend*) können Sie auch aufsteigend sortieren. Sortiert wird nach dem ersten Feld, hier also *artikel*. Normalerweise macht das Sortieren nicht

viel Sinn. Selten wird die gesamte Datenbank ausgegeben, und wenn Sie einen Index nach den Anfangsbuchstaben aufgebaut haben, ist eine einfache Selektion schneller. Das Sortieren großer Datenbanken beansprucht den Server stark. Sie können nur nach den Spalten sortieren, die auch ausgegeben werden.

Nur eindeutige Datensätze ermitteln: DISTINCT

Manchmal sind Datensätze oder Teile davon doppelt. So gibt es in der Tabelle *phone* viele Telefonnummern mehrfach. Wenn Sie alle Personen anrufen möchten, brauchen Sie dieselbe Telefonnummer nicht mehrfach in der Ausgabeliste. Das Schlüsselwort **DISTINCT** hilft weiter. Das folgende Beispiel zeigt alle eindeutigen Autorennamen an:

```
SELECT DISTINCT au_lname FROM address
```

Auch dieser Befehl braucht mehr Leistung von der Datenbankmaschine. Setzen Sie ihn nicht in häufig generierten Abfragen ein.

Auswahl mit Platzhaltern: WHERE LIKE

In den meisten Fällen werden Zeichenketten (*strings*) verarbeitet. SQL unterstützt die Arbeit mit Zeichenketten deshalb sehr komfortabel. So ist es leicht möglich, einen Teilstring zur Selektierung anzugeben, der in allen Feldern an allen Positionen getestet wird. Das Schlüsselwort dafür ist **LIKE**, der folgende Befehl sucht alle Personen, die in Berlin wohnen, egal ob der Name eines Bezirks angehängt wurde (Berlin-Kreuzberg, Berlin-Mitte) oder nicht:

```
SELECT fname FROM address WHERE city LIKE "%Berlin%"
```

Dabei wird hinter das Schlüsselwort **LIKE** der Suchstring gestellt. Die Prozentzeichen sind Platzhalter (engl. wildcards, joker), die in diesem Fall beliebige andere Zeichen vor und nach dem Wort »warm« zulassen. Alle Reisen in warme Länder sollten angezeigt werden.



Hinweis

Eine der häufigsten Fehlerquellen in SQL ist die Verwechslung der Platzhalter mit den von Linux oder Windows bekannten, »*« und »?«. Achten Sie unbedingt darauf, welche Platzhalterzeichen wo verwendet werden.



Tipp

Wenn Sie nach dem Sonderzeichen »%« suchen müssen, können Sie dieses Zeichen so verpacken: `%%`. Für komplexere Anfragen können Sie auch reguläre Ausdrücke verwenden. Dies wird im nächsten Abschnitt gezeigt.

Gruppierungen: GROUP BY

Sie werden oft feststellen, dass Zwischensummen benötigt werden. **GROUP BY** SQL bietet dafür einige Ergänzungen des SELECT-Befehls. Wenn Sie sich beispielsweise die Titel und die dazu passenden Umsätze mit

```
SELECT pub_id, ytd_sales FROM titles
```

ansetzen, erhalten Sie eine lange und wenig hilfreiche Liste. Es ist naheliegend, für die Zusammenfassung der Umsätze der gehandelten Verlage die Summenfunktion heranzuziehen. Da Sie aber in der Liste mehrere Verlage haben, ist eine Zusammenfassung sinnvoll. Dies wird mit GROUP BY erreicht:

```
SELECT pub_id, total = SUM(ytd_sales)
FROM titles
GROUP BY pub_id
```

Das Ergebnis ist deutlich einfacher weiterzuverarbeiten. Die Warnung am Ende deutet darauf hin, dass weitere Reihen vorhanden waren, die wegen enthaltener NULL-Werte nicht beachtet wurden:

Sie können die Ausgabe aber mit dem Schlüsselwort ALL erzwingen.

ALL

Gruppierungen können im Detail komplizierter sein als dieses Beispiel vermuten lässt. Wenn Sie beispielsweise die Anzahl der Titel der Verlage und die Titelnamen anzeigen möchten, wäre folgende Konstruktion naheliegend:

```
SELECT pub_id, title, count(*)
FROM titles
GROUP BY pub_id
```

Dies führt zu einer Fehlermeldung! Der Fehler liegt in der Angabe der Spalte *title*. Die Gruppierung bezieht sich auf die Spalte *pub_id*. Jeder Datensatz, der in der Ausgabeliste angezeigt wird, muss eindeutig sein. Die Anzahl der Titel ist aber uneindeutig (nämlich größer als die Anzahl der Verlage). Welchen Verlag sollte SQL nun pro Datensatz der Ausgabeliste anzeigen?

Die folgende Konstruktion ist dagegen korrekt:

```
SELECT pub_id, max(title), count(*)
FROM titles
GROUP BY pub_id
```

Hier wird die Ausgabe der Namen auf einen bestimmten Namen (nämlich auf den ersten, also nach dem Alphabet) eingeschränkt. Andersherum können Sie auch die Gruppierung auf mehrere Spalten ausdehnen. Stimmen alle Spalten mit allen Gruppen überein, ist die Ausgabe wieder ungruppiert. Hier die andere Variante:

```
SELECT pub_id, title, count(*)  
FROM titles  
GROUP BY pub_id, title
```

Die Funktion COUNT(*) bezieht sich bei Gruppen immer auf die Anzahl der Datensätze in der Gruppe. Die Ausgabe von 1 in der rechten Spalte im letzten Beispiel deutet die Sinnlosigkeit des Unterfangens an.

Gruppierungen einschränken: HAVING

Um mit Gruppen flexibel umgehen zu können, kann auch die Bildung von Gruppen Einschränkungen unterworfen werden. Ähnlich der WHERE-Bedingung können Sie hinter dem Schlüsselwort GROUP BY weitere Bedingungen folgen lassen. Die Bedingung wird gegen das Gruppenergebnis getestet:

```
SELECT pub_id, total = SUM(ytd_sales), COUNT(*)  
FROM titles  
GROUP BY pub_id  
HAVING COUNT(*) > 5
```

Hier werden nur die Verlage angezeigt, die mehr als fünf Veröffentlichungen haben. Zugleich werden die Umsatzsummen ausgegeben.

Solche Auswahlmöglichkeiten können oft auch mit der WHERE-Bedingung erstellt werden. Der einzige (aber oft entscheidende) Unterschied besteht in der Performance. Das erste Beispiel ist deutlich schneller. Die Bildung von Gruppen kostet mehr Rechenleistung als die Selektion nach Kriterien. Wenn Sie also zuerst selektieren (mit WHERE) und dann gruppieren (mit GROUP BY), steigern Sie die Leistung des Gesamtsystems. Der Flexibilität der WHERE-Bedingung kommt nicht zuletzt deshalb große Bedeutung zu. Der folgende Abschnitt zeigt, was Sie mit WHERE noch machen können.

Anzahl der Datensätze begrenzen: LIMIT

Ein weiterer Operator für die Erstellung von Berichten ist LIMIT. LIMIT liefert eine Anzahl von Reihen aus einer größeren Auswahl zurück. So bietet der folgende Befehl nur die ersten 10 Datensätze an:

```
SELECT sname FROM address LIMIT 10
```

Die nächste Variante gibt den sechsten bis elften Datensatz aus:

```
SELECT sname FROM address LIMIT 5,6
```

Beachten Sie, dass die interne Zählung der Reihen bei 0 beginnt.

Erweiterte Bedingungen mit WHERE

AND

Sie können mit WHERE logische Bedingungen angeben. Wollen Sie ein bestimmtes Preisfenster ermitteln, was beispielsweise bei den belie-

ten Online-Reisebüros angeboten wird, eignet sich die folgende Abfrage:

```
SELECT price FROM titles WHERE price>10 AND price<50
```

SQL als Programmiersprache ist relativ gut lesbar. Sehen Sie sich den folgenden Befehl an, der die in anderen Sprachen üblichen logischen Konstruktionen stark vereinfacht: **BETWEEN**

```
SELECT price FROM titles WHERE price BETWEEN 10 AND 50
```

Die Schreibweise mit BETWEEN (dt. *zwischen*) ist zu der oben genutzten Form äquivalent; das Ergebnis ist identisch.

Als Nächstes wollen Sie eine bestimmte Reise und eine Alternative auswählen. Die einfache Abfrage lautet: **OR**

```
SELECT title FROM titles
WHERE type='psychology' OR type='business'
```

Alle Reihen werden ermittelt, die im Feld *title* die Zeichenkette »psychology« oder »business« haben. Gleichwertig ist der folgende Befehl: **IN**

```
SELECT title FROM titles WHERE type IN ('psychology','business')
```

Um die Auswahl zu negieren, ist das Schlüsselwort NOT erlaubt. Bei logischen Bedingungen mit AND und OR kennen Sie das schon vom PHP-Befehl if. Auch die Schlüsselworte BETWEEN und IN lassen sich mit NOT kombinieren.

Die Anwendung ist relativ einfach. Sie können jede logische Operation mit NOT umkehren. NOT verlangsamt aber die Ausführung und sollte nur eingesetzt werden, wenn es keine anderen Konstruktion für den logischen Ausdruck gibt. Testen Sie die folgenden Beispiele mit verschiedenen Werten: **NOT IN und NOT BETWEEN**

```
SELECT title FROM titles WHERE type NOT IN ('business')
```

```
SELECT price FROM titles WHERE price NOT BETWEEN 0 AND 20
```

In manchen Fällen soll eine Auswahl in Abhängigkeit von bestimmten anderen Daten getroffen werden, ohne dass die zur Auswahl verwendeten Daten selbst in das Ergebnis mit einbezogen werden. Diesen Anspruch erfüllt der Operator EXISTS, der TRUE wird, wenn die entsprechende Auswahl gültige Resultate liefert. Das folgende Beispiel ermittelt die Namen von Kunden, die Bestellungen aufgegeben haben. Im Gegensatz zu den bisherigen Beispielen interessieren die Bestellungen selbst hierbei nicht: **EXISTS und NOT EXISTS**

```
SELECT DISTINCT pub_name
FROM publishers
WHERE EXISTS
(SELECT *
```



```
FROM titles
WHERE pub_id = publishers.pub_id AND type = 'business')
```

In diesem Fall wird zuerst der innere SELECT-Befehl abgearbeitet und alle Publikationsnummern aus der Tabelle *titles* werden gesammelt. Diese Liste wird dann mit dem Operator EXISTS auf die Tabelle *publishers* angewandt und die Namen der Titel werden ausgegeben. NOT EXISTS dreht die Bedeutung um, entsprechend zeigt der folgende Befehl alle Kunden, die noch nichts bestellt haben:

```
SELECT DISTINCT pub_name
FROM publishers
WHERE NOT EXISTS
(SELECT *
FROM titles
WHERE pub_id = publishers.pub_id AND type = 'business')
```

INTERVAL

Die Intervall-Funktion prüft eine Zahl N gegen eine Reihe von Zahlen N1..Nn:

```
SELECT INTERVAL(N, N1, N2, N3, ..., Nn)
```

Die Funktion gibt 0 zurück, wenn $N < N_1$, 1 bei $N < N_2$ usw. Es ist notwendig, dass die Prüfwerte in Reihe angegeben werden: $N_1 < N_2 < N_3$.

Reguläre Ausdrücke verwenden: REGEXP und RLIKE

MySQL beherrscht, im Gegensatz zu vielen anderen SQL-Datenbanken, reguläre Ausdrücke. Aus PHP kennen Sie reguläre Ausdrücke bereits. Die Syntax entspricht in jedem Fall dem POSIX 1003.2-Standard.

REGEXP RLIKE

Bei der Anwendung sind die Sonderzeichen gemäß C-Syntax mit Escape-Zeichen zu versehen, der Zeilenumbruch wird also mit \n dargestellt. Mehr Informationen zu regulären Ausdrücken finden Sie in ➔ Abschnitt 3.2.12 *Reguläre Ausdrücke* ab Seite 217. Folgende Syntax verwendet REGEXP:

```
SELECT suchwort REGEXP "Suchmuster"
```

RLIKE dagegen wird zusammen mit WHERE eingesetzt:

```
SELECT * FROM tabelle WHERE field RLIKE "Suchmuster"
```

Zeichenkettenvergleiche: STRCMP

STRCMP

Zeichenkettenvergleiche können Sie auch mit der Funktion STRCMP ausführen. Die Funktion gibt 0 zurück, wenn zwei Zeichenketten gleich sind. Ansonsten wird -1 zurückgegeben, wenn die erste Zeichenkette kleiner als die zweite ist, 1, wenn sie größer ist. Die Entscheidung basiert auf dem für das Sortieren angewendeten Algorithmus.

6.4.8 Operatoren und Funktionen

Beachten Sie, dass die Schreibweise und Symbolik der Operatoren von der in PHP verwendeten teilweise abweicht. Viele Programmierfehler sind auf Verwechslungen von SQL- und PHP-Operatoren zurückzuführen.



Mathematische Operatoren

MySQL beherrscht die folgenden mathematischen Operatoren, die im SELECT-Befehl eingesetzt werden können:

Operator	Beschreibung
+	Addition
-	Subtraktion
*	Multiplikation
/	Division; die Division durch 0 gibt NULL als Ergebnis zurück.

Tabelle 6.4:
Mathematische
Operatoren

Die folgenden Operatoren können auf alle Felder mit dem Datentyp INT und deren Variationen angewendet werden:

Bitoperator	Beschreibung
	Bitweises ODER
&	Bitweises UND
$x \ll n$	x links Verschieben um n Bits.
$x \gg n$	x rechts Verschieben um n Bits.
BIT_COUNT(n)	Anzahl der Bits des Arguments n.

Tabelle 6.5:
Bitoperatoren in
MySQL

Logische Operatoren

Bei Vergleichen oder der Auswahl von Feldern kommen folgende logische Operatoren zum Einsatz:

Operator	Beschreibung
!, NOT	Logische Negation
, OR	Logisches ODER. Der Ausdruck wird Wahr, wenn die linke oder die rechte Seite Wahr ist.
&&, AND	Logisches AND. Der Ausdruck wird Wahr, wenn die linke und die rechte Seite Wahr sind.

Tabelle 6.6:
Logische Operatoren

Vergleichsoperatoren

Vergleichsoperatoren werden zur Bildung logischer Ausdrücke verwendet. MySQL kennt folgende Operatoren:

Tabelle 6.7:
Vergleichsoperatoren

Operator	Beschreibung
=	Gleich
<>	Ungleich
!=	Ungleich
<=	Kleiner als oder Gleich
>=	Größer als oder Gleich
<	Kleiner als
>	Größer als

Mathematische Funktionen

SQL kennt eine ganze Reihe von mathematischen Funktionen, die hier nur als Überblick dargestellt werden sollen. Die exakte Syntax finden Sie im MySQL-Handbuch. Der Einsatz ist unter Umständen schneller und effizienter als bei der Nutzung von PHP.

Tabelle 6.8:
Mathematische
Funktionen

Funktion	Beschreibung
ABS(x)	Absoluter Betrag der Zahl x
ACOS(x)	Arcuskosinus
ASIN(num)	Arcussinus
ATAN(num)	Arcustangens
ATN2(num1, num2)	Arcustangens (Winkel) zwischen num1 und num2
CEILING(x)	Die kleinste Ganzzahl größer oder gleich dem angegebenen numerischen Ausdruck
COS(x)	Kosinus
COT(x)	Kotangens
DEGREES(x)	Umrechnung Radiant in Grad
EXP(x)	Potenz zur Basis e
FLOOR(x)	Die größte Ganzzahl kleiner oder gleich dem angegebenen numerischen Ausdruck.
GREATEST(x1,x2,...)	Gibt den größten Wert der Liste zurück.
LEAST(x1,x2,...)	Gibt den kleinsten Wert der Liste zurück.
LOG(x)	Natürlicher Logarithmus

Funktion	Beschreibung
LOG10(<i>x</i>)	Dekadischer Logarithmus
MOD(<i>n</i> , <i>m</i>)	Modulus, Rest einer Ganzzahldivision
PI()	Konstante π
POWER(<i>x</i> , <i>y</i>)	Potenz
RADIANS(<i>x</i>)	Umrechnung in Radiant
RAND(<i>x</i>)	Zufallszahl, <i>x</i> ist optional
ROUND(<i>x</i> , <i>d</i>)	Rundet Werte mathematisch, die Angabe der Stellen <i>d</i> ist optional.
SIGN(<i>x</i>)	Vorzeichen (gibt -1, 0 oder 1 zurück)
SIN(<i>x</i>)	Sinus
SQRT(<i>x</i>)	Quadratwurzel
TAN(<i>x</i>)	Tangens
TRUNCATE(<i>x</i> , <i>d</i>)	Gibt die Zahl <i>x</i> zurück, gekürzt auf <i>d</i> Dezimalstellen.

Anwendung

Alle mathematischen Funktionen können mit der folgenden Syntax angewandt werden:

```
SELECT FUNCTION(parameters)
```

Ersetzen Sie dabei *FUNCTION(parameters)* durch den Namen der Funktionen und einen numerischen Ausdruck, der den folgenden Datentypen entsprechen muss:

- DEZIMAL, NUMERIC, INT, FLOAT, REAL, TINYINT, MEDIUMINT, BIGINT, SMALLINT

Die Berechnung der Quadratwurzel von 2 schreiben Sie folgendermaßen:

```
SELECT SQRT(2)
```

Logische Funktionen

Diese Funktionen dienen der Steuerung logischer Ausdrücke oder der Abfrage bestimmter Zustände.

- ISNULL(*expr*)

Wertet den Ausdruck *expr* aus und gibt 1 zurück, wenn der Ausdruck NULL ist, sonst 0.

ISNULL()
IFNULL()
IF()



V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

- IFNULL(expr1, expr2)

Wertet den Ausdruck *expr1* aus und gibt *expr2* zurück, wenn der Ausdruck NULL ist, sonst *expr1*.

- IF(expr1, expr2, expr3)

Wenn der Ausdruck *expr1* Wahr ist, wird *expr2* zurückgegeben, sonst *expr3*.

Aggregat-Funktionen

Mit fünf einfachen Rechenoperationen kann auch spaltenweise gearbeitet werden. Die Auswahl beginnt wieder mit dem Schlüsselwort SELECT, dem eine Funktion nachgestellt wird. Die Anwendung ist nur auf numerische Felder sinnvoll.

Tabelle 6.9:
Übersicht über die
Aggregat-Funktionen

Funktion	Beschreibung
AVG	<i>Average</i> . Der Durchschnitt der Felder.
COUNT	<i>Count</i> . Die Anzahl der Felder.
COUNT (*)	Repräsentiert die Anzahl aller Datensätze einer Tabelle.
SUM	<i>Summary</i> . Die Summe der Felder (Addition).
MAX	<i>Maximum</i> . Das Feld mit dem größten Wert bestimmt das Ergebnis.
MIN	<i>Minimum</i> . Das Feld mit dem kleinsten Wert bestimmt das Ergebnis.
STD STDDEV	Statistische <i>Standardabweichung</i> aller Werte der Liste
BIT_OR	Bitweises <i>Oder</i>
BIT_AND	Bitweises <i>Und</i>

Bei den statistischen Funktionen werden NULL-Werte ignoriert. Unterabfragen und Verschachtelungen sind nicht zulässig.

COUNT

Mit am häufigsten werden Sie COUNT verwenden und damit den Wert erzeugen, der in den Skripten dann beispielsweise Schleifen steuert. Der folgende Befehl zeigt an, wie viele verschiedene Wohnorte die Autoren haben:

```
SELECT COUNT(DISTINCT city) FROM authors
```

Warum wird ein Feld ohne WHERE-Bedingung angegeben? SQL gibt hier die Anzahl der Datensätze zurück, die im Feld *city* einen Eintrag haben. Nicht gezählt werden Datensätze, deren Feld *city* den Wert NULL enthält.

Sie können natürlich auch alle Reihen zählen, auch die mit NULL-Werten. Der folgende Befehl zeigt die Anzahl der Buchtitel an: **COUNT(*)**

```
SELECT COUNT(*) FROM titles
```

Oder Sie ermitteln die Anzahl der Reihen, die einer bestimmten Bedingung gehorchen:

```
SELECT COUNT(price) FROM titles WHERE price > 10
```

Ganz ähnlich funktioniert die Bildung des Durchschnittswertes. In diesem Beispiel werden der durchschnittliche Vorschuss und die Summe der Verkaufszahlen des laufenden Jahres für alle Computerbücher berechnet. Beide Aggregat-Funktionen stellen jeweils einen zusammenfassenden Wert für alle gelesenen Zeilen bereit. **AVG, SUM**

```
SELECT AVG(advance), SUM(ytd_sales)
FROM titles
WHERE title LIKE '%computer%'
```

Die Bestellungen mit der kleinsten und größten Bestellsumme sind ebenso leicht zu ermitteln: **MIN, MAX**

```
SELECT MIN(price) FROM titles
```

```
SELECT MAX(price) FROM titles
```

Alle SELECT-Befehle mit Rechenfunktionen lassen sich natürlich um die Bedingung WHERE ergänzen. Im folgenden Abschnitt wird WHERE ausführlich vorgestellt.

System- und Hilfsfunktionen

Einige wichtige Funktionen mit Angaben über Systemzustände und Hilfsfunktionen für Sicherheitsanwendungen finden Sie in Tabelle 6.10.

Funktion	Beschreibung
DATABASE()	Gibt den Namen der aktuellen Datenbank aus.
USER() SYSTEM_USER() SESSION_USER()	Aktueller MySQL-Nutzername
PASSWORD(str)	Erzeugt ein Kennwort zur Zeichenkette <i>str</i> .
ENCRYPT(str, seed)	Erzeugt ein Kennwort zur Zeichenkette <i>str</i> und mit dem Startwert <i>seed</i> . Nutzt das Unix-Kommando crypt. Unter Windows wird NULL zurückgegeben.
LAST_INSERT_ID()	Gibt den zuletzt erzeugten Wert einer AUTO_INCREMENT-Spalte zurück.

Tabelle 6.10:
Die System-
funktionen

Funktion	Beschreibung
FORMAT(<i>n</i> , <i>d</i>)	Formatiert eine Zahl <i>n</i> mit Kommas als Tausendergruppensymbol und Punkt als Dezimaltrennzeichen mit <i>d</i> Dezimalstellen.
VERSION()	Gibt die Versionsnummer des MySQL-Server an.
GET_LOCK(<i>str</i> , <i>to</i>)	Erzeugt eine Verriegelung (Lock) mit dem Namen <i>str</i> und dem Zeitüberschreitungswert <i>to</i> .
RELEASE_LOCK(<i>str</i>)	Gibt die Verriegelung <i>str</i> wieder frei.

Zeichenkettenfunktionen

Zeichenkettenfunktionen dienen der Behandlung von Feldern der Datentypen CHAR und VARCHAR. Bei Operationen mit Zeichenketten, in denen die Position eines Zeichens bestimmt wird, hat die erste Position die Nummer 1.

Tabelle 6.11:
Zeichenketten-
funktionen

Funktion	Beschreibung
ASCII(<i>str</i>)	Gibt den ASCII-Code des Zeichens <i>str</i> zurück. Hat <i>str</i> mehr als ein Zeichen, wird nur das erste Zeichen überprüft.
CHAR(<i>n</i> ,...)	Wandelt die Zahlen <i>n</i> in die entsprechenden ASCII-Zeichen um. Mehrere Argumente werden zu einer Zeichenkette kombiniert.
CONCAT(<i>str</i> ,...)	Verknüpft alle Argumente zu einer Zeichenkette. Wenn eines der Argumente NULL ist, wird NULL zurückgegeben.
LENGTH(<i>str</i>)	Länge der Zeichenketten <i>str</i> .
LOCATE(<i>sub</i> , <i>str</i>) POSITION(<i>sub</i> IN <i>str</i>)	Bestimmt die Position der Zeichenkette <i>sub</i> in der Zeichenkette <i>str</i> .
INSTR(<i>str</i> , <i>sub</i>)	Entspricht LOCATE, nur die Argumente sind vertauscht.
LPAD(<i>str</i> , <i>len</i> , <i>pad</i>)	Fügt <i>pad</i> links an <i>str</i> an, gibt jedoch nur <i>len</i> Zeichen zurück.
RPAD(<i>str</i> , <i>len</i> , <i>pad</i>)	Fügt <i>pad</i> rechts an <i>str</i> an, gibt jedoch nur <i>len</i> Zeichen zurück.
LEFT(<i>str</i> , <i>len</i>)	Gibt <i>len</i> Zeichen vom linken Ende der Zeichenkette <i>str</i> zurück.
RIGHT(<i>str</i> , <i>len</i>)	Gibt <i>len</i> Zeichen vom rechten Ende der Zeichenkette <i>str</i> zurück.

Funktion	Beschreibung
MID(str, pos, len)	Gibt <i>len</i> Zeichen der Zeichenkette <i>str</i> von Position <i>pos</i> an zurück.
SUBSTRING(st FROM 1n) SUBSTRING(st,pos,ln) SUBSTRING(st FROM pos FOR ln)	Andere Schreibweisen für RIGHT und MID. <i>st</i> ist die Zeichenkette, <i>ln</i> die Länge und <i>pos</i> die Startposition.
SUBSTRING(str, pos)	Gibt Teile von <i>str</i> ab Position <i>pos</i> zurück.
SUBSTRING_INDEX(str, delimiter, count)	Gibt den linken Teil einer Zeichenkette zurück, nachdem das Zeichen <i>delimiter</i> sooft aufgetreten ist, wie <i>count</i> angibt.
LTRIM(str)	Entfernt Leerzeichen vom linken Ende.
RTRIM(str)	Entfernt Leerzeichen vom rechten Ende.
TRIM(str)	Entfernt Leerzeichen von beiden Enden der Zeichenkette <i>str</i> .
TRIM BOTH LEADING TRAILING rem FROM str	BOTH entspricht TRIM, LEADING entspricht LTRIM, TRAILING entspricht RTRIM, FROM ist optional, <i>rem</i> ist optional und steht für das zu entfernende Zeichen, <i>str</i> wird bearbeitet.
SOUNDEX(str)	Gibt die Lautfolge für <i>str</i> zurück.
SPACE(n)	Gibt <i>n</i> Leerzeichen zurück.
REPLACE(str,from,to)	Ersetzt alle Vorkommen von <i>from</i> in der Zeichenkette <i>str</i> durch <i>to</i> .
REPEAT(str, count)	Wiederholt die Zeichenkette <i>str</i> <i>count</i> mal.
REVERSE(str)	Dreht eine Zeichenkette um.
INSERT(str,st,len,new)	Fügt <i>len</i> Zeichen der Zeichenkette <i>new</i> an der Stelle <i>st</i> der Zeichenkette <i>str</i> ein.
ELT(n,str1,str2,...)	Gibt die durch <i>n</i> bezeichnete Zeichenkette zurück: <i>str1</i> , wenn <i>n</i> =1 usw.
FIELD(str,str1,str2..)	Gibt die Position von <i>str</i> in <i>str1</i> , <i>str2</i> usw. zurück: Wenn <i>str2</i> = <i>str</i> , wird 2 zurückgegeben.
FIND_IN_SET(str,list)	Gibt die Position von <i>str</i> in der Liste <i>list</i> zurück. Die Liste besteht aus komma-separierten Werten.
MAKE_SET(bits,list)	Wählt die Elemente der Liste anhand der gesetzten Bits in <i>bits</i> aus.
LCASE, LOWER(str)	Wandelt in Kleinbuchstaben um.

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Tabelle 6.12:
Zeichenketten-
funktionen für
Zahlen

Funktion	Beschreibung
UCASE, UPPER(str)	Wandelt in Großbuchstaben um.
Funktion	Beschreibung
CONV(n, from, to)	Konvertiert Zahlen zwischen verschiedenen Zahlenbasen. Zurückgegeben wird immer eine Zeichenkette mit der ermittelten Zahl. <i>n</i> ist die Zahl, <i>from</i> die ursprüngliche Zahlenbasis, <i>to</i> die Zielbasis.
BIN(n)	Gibt eine Zahl <i>n</i> als Zeichenkette im Binärformat zurück, BIN(7) → "111".
OCT(n)	Gibt eine Zahl <i>n</i> als Zeichenkette im Oktalformat zurück.
HEX(n)	Gibt eine Zahl <i>n</i> als Zeichenkette im Hexadezimalformat zurück.



Hinweis

Die Umwandlung von Zeichenketten in Zahlen und umgekehrt wird in MySQL automatisch durchgeführt, wenn dies möglich ist.

Datums- und Zeitfunktionen

Die Datums- und Zeitfunktionen dienen der Ausführung von Umwandlungen und Berechnungen mit Feldern der Datentypen DATE, YEAR, TIME, DATETIME.

Tabelle 6.13:
Datums- und
Zeitfunktionen

Funktion	Beschreibung
DAYOFWEEK(date)	Tag der Woche, 1 ist Sonntag usw. gemäß ODBC-Standard (1 – 7).
WEEKDAY(date)	Tag der Woche, 0 ist Montag (0 – 6)
DAYOFMONTH(date)	Tag des Monats (1 – 31)
DAYOFYEAR(date)	Tag des Jahres (1 – 366)
MONTH(date)	Monat (1 – 12)
DAYNAME(date)	Wochentag (englisch, ausgeschrieben)
MONTHNAME(date)	Monat (englisch, ausgeschrieben)
QUARTER(date)	Quartal (1 – 4)
WEEK(date) WEEK(date, first)	Woche im Jahr (0 – 52). Das optionale Argument <i>first</i> bestimmt, welcher Wochentag als Beginn gezählt wird. 0=Sonntag.
YEAR(date)	Das Jahr vierstellig (1000 – 9999)

Funktion	Beschreibung
HOURL(time)	Stunde (0 – 23)
MINUTE(time)	Minute (0 – 59)
SECOND(time)	Sekunde (0 – 59)
PERIOD_ADD(p,n)	Addiert <i>n</i> Monate zur Periode <i>p</i> . Die Periode wird im Format YYYYMM oder YYMM erwartet. Zurückgegeben wird immer die Langform YYYYMM.
PERIOD_DIFF(p1,p2)	Gibt die Differenz in Monaten zwischen <i>p1</i> und <i>p2</i> zurück
TO_DAYS(date)	Zahl der Tage seit dem Jahr 0
FROM_DAYS(dn)	Ermittelt ein Datum aus der Tageszahl <i>dn</i> .
CURDATE() CURRENT_DATE	Das aktuelle Datum (Systemzeit des Servers)
CURTIME() CURRENT_TIME	Die aktuelle Zeit (Systemzeit des Servers)
NOW() SYSDATE() CURRENT_TIMESTAMP	Datum und Uhrzeit (Systemzeit des Servers)
UNIX_TIMESTAMP	Unix Timestamp (Sekunden in GMT seit dem 1.1.1970, 0 Uhr). Die Funktion wird auch von der Windows-Version unterstützt.
FROM_UNIXTIME(stp)	Gibt ein Datum entsprechend dem Unix Timestamp <i>stp</i> zurück.
FROM_UNIXTIME (stp, format)	Gibt ein Datum entsprechend dem Unix Timestamp <i>stp</i> zurück. Das Datum ist entsprechend <i>format</i> formatiert. Formatangaben entnehmen Sie Tabelle 6.15.
SEC_TO_TIME(sec)	Rechnet die Angabe in Sekunden in das Format HH:MM:SS um.
TIME_TO_SEC(time)	Rechnet eine Zeitangabe in Sekunden um.

Einige Funktionen können eine umfangreiche Datumsarithmetik ausführen. Dazu gehören:

DATE_ADD(date, INTERVAL expr TYPE)

DATE_SUB(date, INTERVAL expr TYPE)

Mit diesen Funktionen wird ein Datum berechnet. DATE_ADD addiert zu *date* Zeiten entsprechend *expr* hinzu. Die folgende Tabelle zeigt, wie sich die Intervallwerte verhalten.

**Datums-
berechnungen**

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Tabelle 6.14:
Zeittypen für
Datumsbe-
rechnungen

TYPE	Bedeutung	Format für expr
SECOND	Sekunde	ss
MINUTE	Minuten	mm
HOURL	Stunde	hh
DAY	Tage	DD
MONTH	Monate	MM
YEAR	Jahre	YY
MINUTE_SECONDS	Minuten und Sekunden	"mm:ss"
HOURL_MINUTE	Stunden und Minuten	"hh:mm"
DAY_HOUR	Tage und Stunden	"DD hh"
YEAR_MONTH	Jahre und Monate	"YY-MM"
HOURL_SECOND	Stunde, Minute, Sekunde	"hh:mm:ss"
DAY_MINUTE	Tag, Stunde, Minute	"DD hh:mm"
DAY_SECONDS	Tag, Stunde, Minute, Sekunde	"DD hh:mm:ss"

Datums- formatierungen

Um die Ausgabe einer Datums- oder Zeitangabe an örtliche Gegebenheiten anzupassen, verwenden Sie die Funktionen zur Datumsformatierung. Zwei Funktionen stehen zur Verfügung:

`DATE_FORMAT(date, format)`

`TIME_FORMAT(time, format)`

format entspricht einer Zeichenkette, die Platzhalter für bestimmte Datumsangaben enthält. Die Platzhalter finden Sie in der folgenden Tabelle:

Tabelle 6.15:
Platzhalter für
Datums-
formatierungen

Platzhalter	Bedeutung (Wertebereich)
%M	Monatsname (January – December)
%W	Wochentagsname (Monday – Sunday)
%D	Monat mit englischem Suffix (1 st , 2 nd , 3 rd usw.)
%Y	Jahr mit 4 Stellen
%y	Jahr mit 2 Stellen
%a	Abgekürzter Wochentag (Mon – Sun)
%d	Tag des Monats (00 – 31) mit führender Null
%e	Tag des Monats (0 – 31) ohne führende Null
%m	Monat (00 – 12) mit führender Null

Platzhalter	Bedeutung (Wertebereich)
%c	Monat (0 – 12) ohne führende Null
%b	Abgekürzter Monatsname (Jan – Dec)
%j	Tag des Jahres (000 – 366)
%H	Stunde (00 – 23) mit führender Null
%k	Stunde (0 – 23) ohne führende Null
%h	Stunde (01 – 12) mit führender Null
%I	Stunde (1 – 12) ohne führende Null
%l	Minuten (0 – 59)
%i	Minuten (00 – 59)
%n	Zeit, 12-Stunden-Format: hh:mm:ss AM PM
%T	Zeit, 24-Stunden-Format: hh:mm:ss
%S	Sekunde (00 – 59)
%s	Sekunde (0 – 59)
%p	AM oder PM
%w	Tag der Woche (0=Sonntag, 6=Samstag)
%U	Woche, Sonntag ist der erste Tag der Woche (00 – 52)
%u	Woche, Montag ist der erste Tag der Woche (00 – 52)
%%	Prozentzeichen

Vor MySQL 3.23 konnten die Formatangaben auch ohne %-Zeichen geschrieben werden. Danach war dies nicht mehr möglich.

6.4.9 Erweiterte SQL-Programmierung

SQL kann mehr, als bisher dargestellt wurde. Bei komplexeren Problemen sind weitere Befehle notwendig, um die Abfragen effektiv ausführen zu können. Dieser Abschnitt stellt solche Techniken vor. Wenn Sie noch tiefer in die Materie einsteigen möchten, ist ein »richtiges« SQL-Buch dringend zu empfehlen.

Verknüpfungen zwischen Tabellen: JOIN

Da jede Person mehrere Telefonnummern haben kann, ist eine Auslagerung in eine weitere Tabelle sinnvoll. Legen Sie eine weitere Tabelle an:

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

```
mysql> CREATE TABLE phone
-> (id INT AUTO_INCREMENT PRIMARY KEY,
->  a_id INT NOT NULL,
->  number VARCHAR(100))
-> ;
Query OK, 0 rows affected (0.02 sec)
```

Um nun die Daten aus beiden Tabellen abzurufen, können Sie die SELECT-Abfrage erweitern:

```
SELECT * FROM address, phone
```

Dies ist nicht das gewünschte Ergebnis, denn die Daten sind bunt gemischt. Zwischen beiden Tabellen muss eine Beziehung hergestellt werden. Man spricht in diesem Fall von einem JOIN.

JOIN

JOIN wird bei verschiedenen SQL-Datenbanken unterschiedlich unterstützt. Es gibt außerdem verschiedene Varianten. Es geht jedoch immer um die Verknüpfung mehrerer Tabellen. Das folgende Syntaxdiagramm zeigt alle in MySQL zulässigen Varianten (jede Zeile stellt eine alternative Schreibweise hinter SELECT ... FROM dar:

Syntax

```
table_reference, table_reference
table_reference [CROSS] JOIN table_reference
table_reference INNER JOIN table_reference join_condition
table_reference STRAIGHT_JOIN table_reference
table_reference LEFT [OUTER] JOIN table_reference join_condition
table_reference LEFT [OUTER] JOIN table_reference
table_reference NATURAL [LEFT [OUTER]] JOIN table_reference
{table_reference LEFT OUTER JOIN table_reference ON condition} }
table_reference RIGHT [OUTER] JOIN table_reference join_condition
table_reference RIGHT [OUTER] JOIN table_reference
table_reference NATURAL [RIGHT [OUTER]] JOIN table_reference
```

Die einfachste Form ist bereits gezeigt worden. Wenn Sie diese Abfrage um WHERE ergänzen, haben Sie einen einfachen funktionierenden JOIN:

```
SELECT * FROM address, phone WHERE a_id = id
```

Hierbei werden die ID-Spalten der beiden Tabellen verglichen und nur die zu den jeweiligen Personen passenden Telefonnummern werden ausgegeben. Aus Gründen der Kompatibilität kann auch das Schlüsselwort JOIN direkt genutzt werden:

```
SELECT * FROM address JOIN phone WHERE a_id = id
```

Man spricht in diesem Fall von einem »Full-Join« – einer vollständigen Verknüpfung zweier Tabellen.

STRAIGHT_JOIN

STRAIGHT_JOIN ist eine MySQL-Erweiterung und arbeitet wie JOIN, allerdings wird hier zuerst die linke Tabelle gelesen und dann die rechte, was unter Umständen zu einer ungünstigen Sortierung führt (JOIN wird intern in der Ausführung optimiert).

LEFT OUTER JOIN kann auch LEFT JOIN geschrieben werden, das Schlüsselwort OUTER existiert nur aus Kompatibilitätsgründen. Bei dieser Form wird immer die komplette linke Tabelle gelesen und bei fehlenden Feldern in der rechten Tabelle werden dort die entsprechenden Reihen mit NULL ergänzt. Auf diese Weise wird sichergestellt, dass auf jeden Fall alle Reihen der linken Tabelle erscheinen. Im Beispiel ist dies praktisch anwendbar, wenn auch die Namen ausgegeben werden sollen, denen keine Telefonnummern zugeordnet wurden. Ein normaler JOIN würde nur Namen anzeigen, denen auch Telefonnummern zugeordnet sind. Anstatt WHERE wird für die Auswahl ON verwendet, die Bedingungen hinter ON entsprechen aber vollständig denen von WHERE.

**LEFT OUTER JOIN
LEFT JOIN**

```
SELECT * FROM address LEFT JOIN phone ON a_id=id
```

LEFT JOIN kann anstatt ON das Schlüsselwort USING verwenden. Sie können hier eine Liste von Spalten aufführen, die in beiden Tabellen enthalten sein müssen. Das folgende Beispiel funktioniert nur, wenn die ID-Spalten in beiden Tabellen denselben Namen tragen:

USING

```
SELECT * FROM address LEFT JOIN phone USING(id)
```

USING(id) entspricht in diesem Fall ON address.id=phone.id. Funktionale Unterschiede gibt es nicht, die Anwendung spart nur Schreibarbeit.

Datensicherung, Im- und Export: LOAD und SELECT INTO

Da die Daten in MySQL in Dateien abgelegt werden, ist eine Datensicherung mit einer Betriebssystemfunktion leicht möglich. Manchmal sollen aber Daten gezielt weitergegeben werden. Mit den Im- und Exportfunktionen können Teile der Datenbank gesichert werden. Außerdem können große Mengen »Fremddaten« ein- oder ausgelagert werden.

Mit LOAD DATA werden Textdateien in eine MySQL-Datenbank importiert. Die vollständige Syntax lautet:

LOAD DATA

```
LOAD DATA [LOCAL] INFILE "filename" [REPLACE|IGNORE]
  INTO TABLE tablename
  [FIELDS
    [TERMINATED BY term]
    [OPTIONALLY] ENCLOSED BY enc1]
    [ESCAPED BY esc ]
  ]
  [LINES TERMINATED BY line]
  [(column, column, ...)]
```

Syntax

Die Textdatei wird, mit vollständigem Pfad, in *filename* eingetragen. Wenn LOCAL angegeben wird, werden die Daten vom Client geholt, sonst vom MySQL-Server. Wenn der Client und der Server über das

Internet verbunden sind, kann der Vorgang erheblich länger dauern. Oft ist es günstiger, die Textdaten per FTP auf den Server zu übertragen und dort ohne die Option LOCAL zu laden. Bei der Angabe der Pfadnamen sind folgende Regeln zu beachten:

- Ein vollständiger (absoluter) Pfad wird komplett ausgewertet.
- Ohne Pfadangabe wird im Verzeichnis der Datenbanken des Servers gelesen.
- Ein unvollständiger (relativer) Pfad wird unterhalb des Datenverzeichnisses gesucht.

Die Pfadangabe IMPORT.TXT sucht daher im Datenbankverzeichnis (kein Pfad). Die Angabe ./IMPORT.TXT sucht dementsprechend im Datenverzeichnis /DATA. Die FIELDS-Auswahl ist optional, ohne Angabe werden die folgenden Standardwerte verwendet:

```
FIELDS TERMINATED BY '\t'
ENCLOSED BY '"'
ESCAPED BY '\\'
```

Die Angaben dienen der Festlegung der Trennzeichen der Textdatei:

- TERMINATED gibt die Feldtrennung an, '\t' steht für den Tabulator. Üblich sind auch Semikola oder Kommata.
- ENCLOSED gibt das Zeichen an, mit dem Zeichenketten umschlossen sind. Dies ist bei kommaseparierten Dateien (CSV) üblich, da Kommata auch in Zeichenketten vorkommen können.
- ESCAPED gibt das Escape-Zeichen an, mit dem Sonderzeichen gekennzeichnet sind. Üblicherweise der Backslash \.

Unabhängig von der Angabe der Feldtrenner kann auch der Zeilenumbruch oder Datensatztrenner eingestellt werden:

```
LINES TERMINATED BY line
```

- line steht für das Trennzeichen, normalerweise '\n' für den Zeilenumbruch.

Beachten Sie, dass diese Angaben beim Speichern mit SELECT INTO übereinstimmen müssen, wenn Sie Daten zur Sicherung mit diesen Funktionen im- und exportieren. Es ist empfehlenswert, sich dazu in PHP zwei zueinander compatible Funktionen zu schreiben, die das zuverlässig erledigen.



Hinweis

Der Import von Textdateien mit Feldern fester Breite (fixed-size) wird nicht unterstützt.

Wenn Sie in eine Tabelle importieren, die schon Daten enthält, können Sie das Überschreiben vorhandener Daten mit REPLACE und IGNORE

steuern. Dies gilt nur für Felder, die UNIQUE sind (eindeutige Werte enthalten).

Der Befehl **SELECT ... INTO OUTFILE** ist komplementär zu **LOAD DATA** und verwendet eine ähnliche Syntax. Sie können damit Daten aus einer MySQL-Datenbank in eine Textdatei exportieren.

```
SELECT expression | (column, column, ...) INTO OUTFILE "filename"
[FIELDS
  [TERMINATED BY term]
  [OPTIONALLY] ENCLOSED BY enc1]
  [ESCAPED BY esc ]
]
[LINES TERMINATED BY line]
FROM tablename
[condition]
```

Syntax

Die Parameter entsprechen den bei **LOAD DATA** beschriebenen. Außerdem entspricht die Auswahl der Felder mit *condition* den bei **SELECT** und **WHERE** beschriebenen.

Systemfunktionen

Einige Funktionen sind für die Stabilität und Leistung des MySQL-Servers von Bedeutung. Sie werden in diesem Abschnitt vorgestellt. Für die ersten Schritte mit MySQL benötigen Sie diese Funktionen nicht, ebenso wenig wie auf einem Entwicklungssystem.



MySQL verwendet intern eine Reihe von Puffern und Caches, um die Leistung zu steigern. Mit **FLUSH** können Sie diese Puffer gezielt leeren.

```
FLUSH option, option
```

Syntax

Die verschiedenen Optionen können kombiniert werden:

- **HOSTS**

Leert die Cache-Tabellen der Hosts. Wenn viele Hosts verbunden sind und häufig deren IP-Nummern wechseln oder Verbindungsfehler auftreten, werden Hosts gesperrt. Mit dieser Option werden die Cache-Tabellen gelöscht und neue Verbindungen erlaubt.

- **LOGS**

Schließt die Protokolldatei und eröffnet eine neue. Wenn nicht mit einer Namenserverweiterung gearbeitet wird, zählt der Befehl die Dateinamen hoch (aus LOG.1 wird LOG.2).

- **PRIVILEGES**

Lädt die Tabelle der Zugriffsberechtigungen neu.

- TABLES

Schließt alle offenen Tabellen.

- STATUS

Setzt die meisten Statusvariablen auf 0.

KILL Jede Verbindung zu MySQL hat eine eindeutige Thread-ID. Mit dem folgenden Befehl kann die Verbindung für diese Thread-ID gezielt unterbrochen werden:

Syntax

KILL id

Alle vorhandenen Threads werden mit `SHOW PROCESSLIST` angezeigt (nicht unter Windows). Allerdings können Sie unter Windows die Prozesse mit dem Werkzeug WinMySQLAdmin einsehen (siehe dazu ➡ Abschnitt 6.2.1 *winMySQLAdmin* ab Seite 460).

SHOW Der Befehl `SHOW` zeigt verschiedene Zustände im System an. Die folgende Auflistung zeigt alle Varianten:

Syntax

SHOW option

Der komplette Befehl kann dabei folgendermaßen aussehen:

- `SHOW DATABASES [LIKE platzhalter]`
Zeigt alle Datenbanken des Servers an.
- `SHOW TABLES [FROM database] [LIKE platzhalter]`
Zeigt die Tabelle der aktuellen oder der ausgewählten Datenbank an.
- `SHOW COLUMNS FROM table [FROM database] [LIKE platzhalter]`
Zeigt die Spalten einschließlich aller Eigenschaften der gewählten Tabelle an.
- `SHOW INDEX FROM table [FROM database]`
Zeigt die Indizes einer Tabelle an.
- `SHOW STATUS`
Zeigt allgemeine Statusinformationen an.
- `SHOW VARIABLES`
Zeigt verschiedene Systemvariablen an.
- `SHOW PROCESSLIST`
Zeigt die Liste aktiver Threads an (dies funktioniert nur unter Unix).

Das Schlüsselwort **EXPLAIN** wird vor ein **SELECT** gesetzt und zeigt an, wie der **SELECT**-Befehl ausgeführt worden *wäre*. Dazu werden verschiedene Parameter ermittelt und in einer Tabellenübersicht angezeigt. Eine ausführliche Beschreibung der angezeigten Werte finden Sie in der Dokumentation.

Der Befehl setzt bestimmte Parameter. Damit kann das Systemverhalten beeinflusst werden. Verwenden Sie folgende Syntax:

SET [**OPTION**] *option* = wert

Syntax

Das Schlüsselwort **OPTION** ist optional und hat keine Funktion außer schön auszusehen. Für *option* können Sie die folgenden Parameter einsetzen:

- **CHARACTER SET** Zeichensatz | **DEFAULT**

Setzt den Zeichensatz neu. In der Standarddistribution ist der einzige zulässige Wert `cp1251_koi8`.

- **PASSWORD** = **PASSWORD**('password')

Setzt das Kennwort für einen nicht anonymen Nutzer. Jeder nicht anonyme Nutzer kann sein eigenes Kennwort ändern.

- **PASSWORD FOR username** = **PASSWORD**('password')

Setzt das Kennwort für einen bestimmten Nutzer. Dies kann nur der Administrator. Der Nutzernamen *username* sollte in dem Format `USER@HOST` angegeben werden.

- **SQL_BIG_TABLES** = 0 | 1

Wird dieser Parameter auf 1 gesetzt, werden Tabellen auf der Festplatte und nicht im Speicher gehalten. Dies verlangsamt die Ausführung, vermeidet aber Fehler beim Anlegen sehr großer temporärer Tabellen.

- **SQL_BIG_SELECTS** = 0 | 1

Wenn der Wert 1 ist, werden **SELECT**-Befehle nicht ausgeführt, die sehr lange dauern würden.

- **SQL_LOW_PRIORITY_UPDATES** = 0 | 1

Wenn 1, werden alle **INSERT**-, **UPDATE**- und **DELETE**-Befehle zurückgehalten, bis kein **SELECT** mehr aktiv ist.

- **SQL_SELECT_LIMIT** = wert | **DEFAULT**

Mit *wert* legen Sie fest, wie viele Datensätze **SELECT** maximal zurückgibt.

- `SQL_LOG_OFF = 0 | 1`

Wenn 1, wird das Standardprotokoll für den Client nicht geschrieben.

- `SQL_LOG_UPDATE = 0 | 1`

Wenn 0, wird kein Update-Protokoll für den Client geschrieben.

- `TIMESTAMP = wert | DEFAULT`

Setzt die Zeit für den Client.

- `LAST_INSERT_ID = #`

Setzt den Wert auf #, den der Befehl `SELECT LAST_INSERT_ID` zurück gibt.

- `INSERT_ID = #`

Setzt den Wert, der beim nächsten Auftreten eines `INSERT` in einer `AUTO_INCREMENT`-Spalte verwendet wird.

SHUTDOWN

Der Befehl `SHUTDOWN` fährt den MySQL-Server herunter. Das betrifft nur den Dienst bzw. Daemon, nicht die Servermaschine.

6.5 MySQL-Sicherheit

Auch der Datenbankserver muss vor möglichen Angriffen geschützt werden. Oft werden zwar Skripte gesichert, der direkte Zugriff per TCP/IP aber nicht unterbunden. Mit der Sicherheit beschäftigt sich dieser Abschnitt.

6.5.1 Einführung

In den bisherigen Einführungen haben Sie sich jeweils direkt mit dem MySQL-Server verbunden. Der Aufruf erfolgte über die MySQL-Konsole. Damit sind Sie als Administrator angemeldet. Der Nutzer, der über das Internet zugreift, darf natürlich keine Administratorrechte haben. Deshalb ist es möglich, verschiedene Nutzer einzurichten und mit spezifischen Rechten zu versehen. Für eine Applikation, die den MySQL-Server direkt nutzt, können sogar viele Nutzer verwaltet werden.

Was gesichert wird

Die Zugriffsrechte regeln die Fähigkeit einzelner Nutzer, bestimmte Datenbanken, Tabellen und Spalten mit den Befehlen `SELECT`, `INSERT`, `DELETE` und `UPDATE` bearbeiten zu können. Welche Rechte Sie konkret jedem Nutzer zugestehen, hängt von Ihrer Applikation ab.

In der Anfangszeit werden komplexe Rechtesysteme eher hinderlich sein. Sie sollten für Webapplikationen eine getrennte Datenbank verwenden und diese global schützen.



Grundsätzlich ist es erlaubt, mit einem anonymen Nutzer zu arbeiten. Dies werden in der Regel all jene Nutzer sein, die über den Webserver, also über PHP-Skripte, zugreifen.

Namen und Kennwörter

Die Zugriffsrechte und die damit verknüpften Namen und Kennwörter in MySQL haben nichts mit den Login-Namen unter Unix oder Windows zu tun. Usernamen dürfen in MySQL 16 Zeichen lang sein. Die Kennwortverschlüsselung ist nicht mit der von Unix verwendeten identisch.

Zu MySQL verbinden

Um sich als Nutzer mit bestimmten Rechten mit dem Server zu verbinden, kann folgender Befehl verwendet werden:

```
$ mysql [-h host] [-u user] [-ppassword]
```

Beachten Sie, dass hinter `-p` kein Leerzeichen steht (bei den anderen Optionen wird ein Leerzeichen erwartet). Wenn Sie einzelne Angaben weglassen, werden folgende Standardwerte eingesetzt:

- `-h`. localhost
- `-u`. Unix-Loginname oder Windows-Loginname
- `-p`. entfällt (kein Kennwort)

Wenn das Kennwort am Prompt eingegeben wird, steht es im Klartext auf dem Bildschirm. Sie können das Kennwort weglassen und nur `-p` schreiben. Dann erfolgt die Eingabe in der Eingabeaufforderung, in der die getippten Zeichen durch Sternchen ersetzt werden.

GRANT und REVOKE

Die SQL-Befehle GRANT und REVOKE steuern die Zugriffsrechte auf Kommandoebene. GRANT gewährt Rechte, REVOKE entfernt sie wieder. Die Befehle stehen ab MySQL 3.22.11 zur Verfügung.

Der Befehl GRANT hat folgende Syntax:

GRANT

```
GRANT priv_type [(column,column,...)] [,[(...)]]
ON { table | * | *.* | db.* }
TO username [IDENTIFIED BY 'password']
[,username [IDENTIFIED BY 'password']]
[WITH GRANT OPTION]
```

Syntax

Der Befehl REVOKE hat folgende Syntax:

REVOKE

Syntax

```
GRANT priv_type [(column,column,...)] [,[(...)]]
ON { table | * | *.* | db.* }
FROM username [,username]
```

Level der Nutzerrechte

Der Umgang mit GRANT und REVOKE erlaubt es Systemadministratoren, Nutzerrechte in vier Stufen einzustellen:

- *Global Level*

Diese Rechte gelten für alle Datenbanken auf dem Server. Sie werden in der Tabelle MySQL.USER gespeichert.

- *Database Level*

Diese Rechte beziehen sich auf eine bestimmte Datenbank. Sie werden in den Tabellen MySQL.DB und MySQL.HOST gespeichert.

- *Table Level*

Diese Rechte steuern den Zugriff auf einzelne Tabellen einer Datenbank. Sie werden in MySQL.TABLES_PRIV gespeichert.

- *Column Level*

Die feinste Stufung unterscheidet Zugriffsrechte auf Spaltenebene. Die Speicherung erfolgt in MySQL.COLUMNS_PRIV.



Hinweis

GRANT und REVOKE stehen in einer geschützten Umgebung, beispielsweise im Webpace eines Providers, nicht zur Verfügung. Der über den Webserver zugreifende User hat selbst nicht die entsprechenden Rechte.

Der Parameter *priv_type* im Syntaxdiagramm bestimmt, welche Kommandos der betroffene Nutzer ausführen darf. Sie können einen oder mehrere der in Tabelle 6.16 genannten Befehle einsetzen. Für Tabellen sind nur die *kursiv* gedruckten, für Spalten sind nur die **fett** gedruckten Befehle zulässig.

Tabelle 6.16:
Rechte, die Usern
erteilt werden können

Befehl	Bedeutung
ALL PRIVILEGES	Alle Rechte
USAGE	Keine Rechte
<i>ALTER</i> , <i>CREATE</i> , <i>DELETE</i> , <i>DROP</i> , <i>FILE</i> , <i>GRANT</i> , <i>INDEX</i> , <i>INSERT</i> , <i>PROCESS</i> , <i>RELOAD</i> , <i>SELECT</i> , <i>SHUTDOWN</i> , <i>UPDATE</i>	Erteilt dem Nutzer das Recht, den genannten Befehl ausführen zu dürfen

Nutzernamen

Nutzernamen müssen in der Form NAME@HOST angegeben werden. Es ist bei GRANT und REVOKE erlaubt, Platzhalterzeichen (%) zu verwenden. Wenn die Namen Platzhalter enthalten, müssen sie in Anführungszeichen gesetzt werden.

Wenn Sie Rechte ändern, müssen Sie das Kommando `FLUSH PRIVILEGES` ablaufen lassen oder den MySQL-Server erneut starten. Die Rechtetabellen werden zur Laufzeit im Speicher gehalten.



6.5.2 Die Usertabellen in MySQL

Die Speicherung der Zugriffsrechte erfolgt in fünf Tabellen der internen Datenbank `MYSQL`: `USER`, `DB`, `HOST`, `TABLES_PRIV`, `COLUMNS_PRIV`. In den Tabellen finden Sie bestimmte Spalten, die jeweils einzelne Zugriffsrechte enthalten. Die folgende Tabelle zeigt, welche Zugriffsrechte in welchem Kontext von Bedeutung sind.

Recht	Spalte	Kontext
SELECT	<i>Select_priv</i>	Tabelle
INSERT	<i>Insert_priv</i>	Tabelle
UPDATE	<i>Update_priv</i>	Tabelle
DELETE	<i>Delete_priv</i>	Tabelle
INDEX	<i>Index_priv</i>	Tabelle
ALTER	<i>Alter_priv</i>	Tabelle
CREATE	<i>Create_priv</i>	Datenbank, Tabelle oder Index
DROP	<i>Drop_priv</i>	Datenbank oder Tabelle
GRANT	<i>Grant_priv</i>	Datenbank oder Tabelle
RELOAD	<i>Reload_priv</i>	Serveradministration
SHUTDOWN	<i>Shutdown_priv</i>	Serveradministration
PROCESS	<i>Process_priv</i>	Serveradministration
FILE	<i>File_priv</i>	Dateizugriff auf den Server

Tabelle 6.17:
Kontext der
Zugriffsrechte

Einige Rechte korrespondieren nicht direkt mit einem SQL-Befehl, sondern mit einer Gruppe. Tabelle 6.18 gibt darüber Auskunft:

Zugriffsrecht	Abhängige Befehle
RELOAD	RELOAD, REFRESH, FLUSH
PROCESS	PROCESSLIST, KILL

Tabelle 6.18:
Zugriffsrechte für
Befehlsgruppen

Besonderheiten der Zugriffsrechte

Bei der Verteilung der Zugriffsrechte sind einige Besonderheiten zu beachten, da sich andernfalls Sicherheitslücken ergeben können.

- Mit dem Zugriffsrecht `GRANT` können Nutzer anderen Nutzern die eigenen Zugriffsrechte erteilen. Zwei Nutzer, die beide `GRANT`-

Rechte haben, können damit ihre Rechte kombinieren und verfügen anschließend über mehr Rechte, als Ihnen ursprünglich zugestanden werden sollte.

- FILE erlaubt das Lesen jeder Datei. Dieses Recht kann missbraucht werden, um Dateien in Tabellen einzulesen und dann mit SELECT (einem geringeren Recht) zur Anzeige zu bringen.
- SHUTDOWN fährt den Server herunter, ohne Rücksicht auf andere Nutzer, die möglicherweise höhere Rechte haben.
- PROCESS kann lesend auf die Konsole zugreifen, an der möglicherweise Kennwörter ungeschützt eingegeben werden.
- Kennwörter können zwar nicht gelesen werden, privilegierte Nutzer können diese aber durch eigene ersetzen.

6.6 ODBC

ODBC ist ein universelles Interface für Datenbanken. Damit erschließen Sie sich bei der PHP-Programmierung alle Datenbanken, die nicht mit nativen Funktionen angesprochen werden können. Die häufigste Anwendung dürfte der Zugriff auf Microsoft Access sein.

6.6.1 Einführung in ODBC

Dieser Abschnitt beschreibt die Grundlagen von ODBC in knapper Form, sodass Sie zügig mit der Implementierung der vorgestellten Funktionen beginnen können.

Was ist ODBC?



ODBC ist eine Implementierung der Firma Microsoft eines Call Level Interface nach dem SQL-Standard. Sie liefert damit eine Standard-Applikations-Schnittstelle, die es einer ganzen Reihe von Clients (beispielsweise Windows-Programme oder Skripte) erlaubt, auf die Ressourcen einer Datenquelle zuzugreifen. SQL-Befehle werden auf drei Niveaus unterstützt. Für eine Applikation muss eine Aussage getroffen werden, welches Niveau zur Ausführung erforderlich ist. Die Datenquelle selbst kann ein SQL-Server sein, ebenso gut wäre auch eine proprietäre Datenbank oder nur eine Textdatei möglich.

ODBC wird inzwischen in der Industrie weitreichend unterstützt. Zwar steht nur unter Windows eine komfortable Benutzerführung zur Verfügung, der Einsatz ist jedoch auch unter Unix möglich.

Vor- und Nachteile

ODBC ist eine universelle Schnittstelle zur Anbindung von Datenbanken. Basiert eine Applikation auf ODBC, kann man in der Regel die dahinter liegende Datenbank austauschen, ohne dass Änderungen am Programm notwendig werden. Bei der Nutzung nativer Treiber für den Datenbankzugriff ist dies nicht immer möglich. Ohne ODBC ist es sehr schwer, datenbankunabhängig zu programmieren. Nahezu jede Datenbank unterstützt ODBC und steht damit überall zur Verfügung.

Vorteile

Die Vorteile proprietärer Funktionen bestimmter Datenbanken, die oft einer enormen Leistungssteigerung dienen, werden durch ODBC natürlich nicht genutzt. Abgesehen von dieser grundsätzlichen Einschränkung ist ODBC verhältnismäßig langsam, denn jeder Befehl durchläuft eine universelle Schnittstelle, umgekehrt gilt dies auch für die von der Datenbank zurückgegebenen Daten. Einige Datenbanken bieten auch ganz spezielle Sicherheitsfeatures; diese lassen sich mit ODBC kaum nutzen.

Nachteile

Auch in Bezug auf die künftige Verfügbarkeit ist ODBC kritisch zu sehen. Microsoft propagiert schon seit einiger Zeit OLE DB als neue universelle Schnittstelle zu Datenbankservern. Diese ist einfacher und effizienter. Ein offener Standard wie ODBC ist sie dagegen nicht. Mit PHP kann OLE DB auf Windows-Plattformen verwendet werden, wenn mit COM gearbeitet wird. Dazu müssen Sie aber entweder auf ADO zurückgreifen, was für PHP-Entwickler eher gewöhnungsbedürftig ist, oder – wenn MySQL eingesetzt wird – auf das ActiveX-Steuerelement von MySQL. Beides sind jedoch eher exotische Anwendungen und dürften selten verwendet werden. Sie werden deshalb hier nicht weiter betrachtet.

Wie wird ODBC genutzt?

ODBC ist im eigentlichen Sinne ein Standard, also kein kaufbares Produkt und auch keine Anwendung. Normalerweise liefern die Hersteller von Datenbanken für alle erdenklichen Betriebssysteme passende ODBC-Treiber. Wenn Sie beispielsweise mit Windows NT arbeiten, können Sie mehrere Datenbanken nutzen, unter anderem:

- Textdateien
- MS Access (alle Versionen von 97 bis 2002)
- MS Excel
- MySQL
- SQL Server (alle Versionen von 6.5 bis 2000)
- Oracle

Wenn Sie nun zugleich Skripte in ASP, Perl und PHP schreiben, kann eine wechselseitige Anbindung der Datenbank aufwändig werden. Mit ODBC ist es einfacher, damit umzugehen.

In PHP wird ODBC durch eine Vielzahl an Funktionen unterstützt. Sie können somit Skripte schreiben, deren einzige Schnittstelle zur Datenbank ODBC ist. Der Vorteil ist, dass Sie auf einem Entwicklungssystem mit einer preiswerten Datenbank wie Access entwickeln, die Produktionsserver aber dann mit Oracle ausstatten können. Änderungen an den Skripten und insbesondere an den Datenbankabfragen sind kaum nötig. Unabhängig davon werden natürlich einige Befehle direkt an die Datenbank weitergeleitet. ODBC verhindert also nicht automatisch, dass Programmierer systemabhängige Befehle nutzen.

ODBC-Architektur

ODBC baut auf einer Vier-Schicht-Architektur auf, die aus folgenden Teilen besteht:

- Applikationsschicht (Ihr PHP-Skript)
- Treiber-Manager (Windows ODBC-Manager)
- ODBC-Treiber (MySQL-ODBC-Treiber für Windows)
- Datenquelle (MySQL-Server)

Die Datenquelle

Datenquelle
Data Source Name
DSN

Der wichtigste Begriff im Zusammenhang mit ODBC ist die Datenquelle (Data Source). Die Datenquelle wird mit einem Namen benannt, dem Data Source Name (DSN). Dies ist ein logischer Name, der frei vergeben werden kann. Der Name wird im ODBC-Manager eingerichtet. Dies wird im folgenden Abschnitt beschrieben.

Es gibt drei Arten von ODBC-Quellen, auf die zugegriffen werden kann:

- *User-DSN*

Nutzerspezifische Datenquellen. Wird nur verwendet, um »private« Datenquellen zu erzeugen. Auf diese Quelle kann nur der lokale Nutzer des Computers zugreifen; die Datenquelle muss auch auf diesem Computer laufen.

- *System-DSN*

Eine Datenquelle, die der Maschine zugeordnet ist. Auf diese Quellen können Personen zugreifen, die Zugriff auf den Computer selbst haben. Auch laufende Dienste, wie der Webserver, können auf diese Datenquellen zugreifen.

- *File-DSN*

Eine nutzerspezifische Datenquelle, die von mehreren Personen genutzt werden kann. Sie kann irgendwo im Netzwerk liegen und alle, die über gleichartige Treiber verfügen, können darauf zugreifen. Diese DSN speichert die Parameter in Textdateien.

Am besten, Sie geben eine System-DSN an. Diese Form ist am einfachsten zu verwalten, der Webserver erhält einen problemlosen Zugriff. Die Einrichtung nehmen Sie in der Systemsteuerung vor.

6.6.2 ODBC unter Windows einrichten

Der folgende Abschnitt ist nur für Windows-Nutzer interessant, die mit PHP auf die dort verbreiteten Datenbanken zugreifen möchten, beispielsweise um alte Datenbestände nach MySQL importieren zu können.



Datenprovider Microsoft SQL Server

Sie finden die Einstellungen zu ODBC in der Systemsteuerung unter dem Icon ODBC. Prinzipiell bedeutet ODBC nicht die völlige Unabhängigkeit von allem, sondern einfach nur ein Treiberkonzept. Sie benötigen ODBC-Treiber für jedes verwendete Betriebssystem und für die entsprechende Datenbank – im Falle des Microsoft SQL Servers ⁷²² also einen dafür passenden Treiber, der unter Windows NT/2000 läuft. Jeder Treiber besteht aus einer DLL (Dynamic Link Library), die im \SYSTEM32-Ordner liegt. Wenn Sie MS SQL Server installieren – und das gilt auch für die meisten anderen SQL-Server –, wird der richtige ODBC-Treiber automatisch mit installiert.



Wählen Sie dazu das ODBC-Icon und dann die Registerkarte SYSTEM DSN aus. Dann wählen Sie die Schaltfläche HINZUFÜGEN..., um eine neue Datenquelle anzulegen.

²² Die Darstellung gilt sinngemäß auch für den SQL Server 2000

Abbildung 6.11:
Eine neue System-
datenquelle wird
angelegt. Vergeben
Sie einen Namen,
eine Beschreibung
und wählen Sie den
SQL-Server aus.



Schreiben Sie in das Dialogfeld MIT WELCHEM SQL SERVER MÖCHTEN SIE SICH VERBINDEN? den Namen Ihres Computers oder der Maschine, auf der der SQL-Server läuft. Klicken Sie dann auf WEITER> (siehe Abbildung 6.11). Im folgenden Dialogfenster wählen Sie erst die Option MIT SQL SERVER AUTHENTIFIZIERUNG... aus. Im unteren Teil des Dialoges (siehe Abbildung 6.12) können Sie dann den Namen des Datenbanknutzers einstellen. Geben Sie hier zuerst nur »sa« ohne Kennwort ein. Damit kann der ODBC-Treiber nun arbeiten; Ihre PHP-Skripte haben Zugriff auf alle Tabellen und Datenbanken des Servers.



Hinweis

Erst wenn Sie als Provider fremden Personen Zugriff auf ASP erlauben, und diese Ihren SQL-Server nutzen, lohnt es sich für jeden Kunden, einen SQL-Server-Login zu vergeben und jede einzelne private Systemdatenquelle darüber zu schützen. Dann werden Sie auch die andere Option MIT WINDOWS NT-AUTHENTIFIZIERUNG... und die Benutzersteuerung von Windows NT/2000 nutzen. Wichtig ist aber, dass diese Einstellungen auch im SQL-Enterprise-Manager vorgenommen werden.

Abbildung 6.12:
Einstellungen für
einen korrekten
Zugriff auf die SQL-
Datenbank



Klicken Sie nun auf WEITER>, wählen Sie im folgenden Dialog die bevorzugte Datenbank aus. Es ist erforderlich, eine eigene Datenbank anzulegen. Wenn Sie das mit dem SQL Enterprise Manager getan haben, wählen Sie die Datenbank einfach aus der Liste aus. Der ODBC-Treiber sollte zu diesem Zeitpunkt bereits Zugriff auf den SQL-

Server haben und alle vorhandenen Datenbanken anzeigen. Stellen Sie die anderen Optionen wie in Abbildung 6.13 ein. Die beiden folgenden Dialogfelder lassen Sie unverändert. Sie können evtl. für die Angabe von Daten und Zeiten die regionalen Einstellungen wählen, dann bestimmt Windows, welche Sprache zugrunde gelegt wird. SQL selbst »spricht« derzeit nur amerikanisches Englisch. Das letzte Feld erlaubt die Angabe von Protokolldateien. Das ist zumindest während der Entwicklungsphase eher hinderlich. Später müssen Sie selbst entscheiden, ob ein SQL-Log sinnvoll ist.

Meine Empfehlung ist, das Protokoll nur dann einzuschalten, wenn Sie auf Fehlersuche sind. Es ist unter Umständen ein effektives Werkzeug. Stabil laufende Systeme benötigen dieses Log nicht.



Hinweis



Abbildung 6.13:
Auswahl der
Datenbank und
einiger Kompa-
tibilitäts-
einstellungen

Nach dem Klick auf den Schalter FERTIGSTELLEN wird die Datenquelle angelegt; Sie können in einem Textfenster den Zugriff darauf testen. Abbildung 6.14 zeigt, wie dies aussehen sollte.

Nun ist es soweit! Sind alle Tests bestanden, und der ODBC-Treiber arbeitet, können Sie mit den ODBC-Funktionen in PHP auf die SQL-Datenbank zugreifen. Das nötige Wissen vermittelt ➔ Abschnitt 7.4 *Zugriff auf ODBC* ab Seite 570.

Abbildung 6.14:
Anzeige der
»gesammelten«
Daten. Wenn Sie
nun auf
DATENQUELLE
TESTEN... klicken,
sollte ein weiteres
Fenster erscheinen
und Erfolg melden.



6.6.3 Datenbankzugriff mit Access

Im Gegensatz zum SQL-Server ist Access einfacher, Sie kommen schneller zu einer funktionierenden Datenbank. Durch die universelle Abfragesprache SQL ist es später leicht möglich, ein mit Access lokal erstelltes Programm auf einen Webserver zu übertragen, der mit einer starken SQL-Datenbank arbeitet. Über ODBC können Sie von PHP aus auf beide Programme zugreifen, ohne sich ein Problem durch die nativen MS SQL-Funktionen einzuhandeln.

Datenbank in Access 97/2000 anlegen

Eine Entwicklungsdatenbank mit Access anlegen

Starten Sie Access und legen Sie mit DATEI | NEUE DATENBANK ANLEGEN... eine neue Datenbank an. Sie können einen Assistenten oder eine der mitgelieferten Vorlagen benutzen. Für die meisten Webprojekte dürfte das nicht so sinnvoll sein. Legen Sie für dieses Beispiel eine leere Datenbank an (Abbildung 6.15).

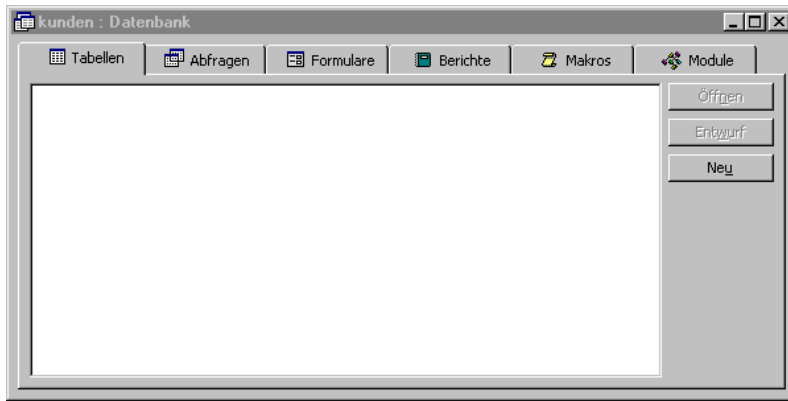


Abbildung 6.15:
Eine leere Datenbank
in Access

Mit der Schaltfläche NEU... können Sie nun eine neue Tabelle anlegen. Access fragt nun nach der Ansicht, mit der die Tabelle erstellt werden soll. Hier ist die Auswahl DATENBANKANSICHT oder ENTWURFSANSICHT interessant, denn Access wird bei den PHP-Skripten nicht als Front-End – als Bedienoberfläche – eingesetzt. Die Gestaltung der Formulare spielt keine Rolle.

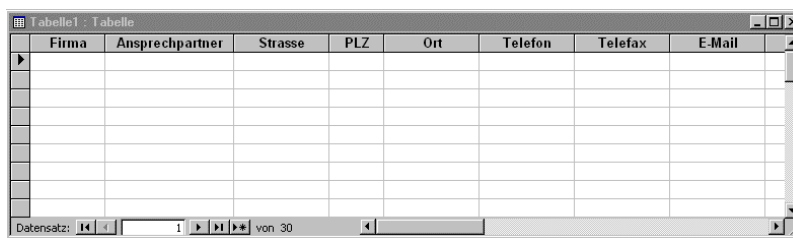


Abbildung 6.16:
Anlegen einer Tabelle
mit Access und
Benennung der
Spalten in der
Datenbankansicht

Um die Spalten in der Datenbankansicht zu benennen, doppelklicken Sie mit der Maus auf die Spaltenüberschriften (Feld1, Feld2 usw.) und tragen dort direkt den Namen ein. Abbildung 6.16 zeigt, wie eine solche Tabelle dann aussieht. Sie können die Feldnamen auch gleich in der Entwurfsansicht eingeben, die weiter unten beschrieben wird.

In die Reihen können Sie nun Testdaten direkt eingeben, die zum Ausprobieren der Skripten benötigt werden. Das ist weitaus bequemer, als über zusätzliche Skripten die Daten aufzufüllen.

Wenn Sie die Tabelle speichern, fragt Access, ob Sie einen Primärschlüssel anlegen wollen. Bestätigen Sie die Frage mit JA, wenn ein solcher Schlüssel benötigt wird. Wenn Sie unsicher sind, ob Ihre Applikation einen Primärschlüssel benötigt, wählen Sie ebenfalls JA.

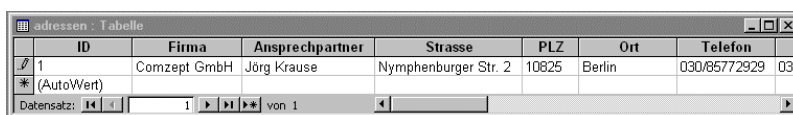
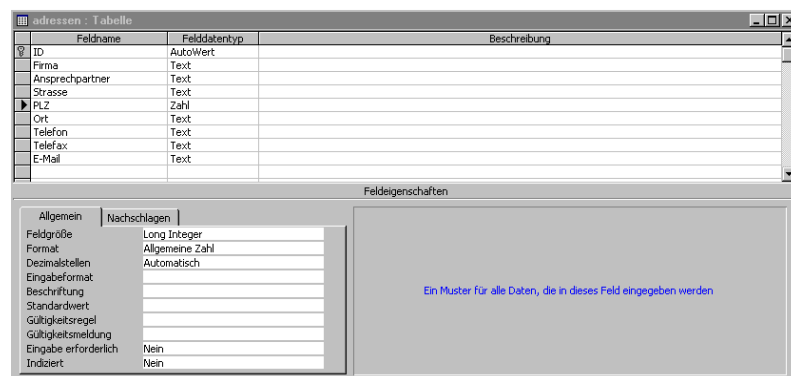


Abbildung 6.17:
Access hat einen
Primärschlüssel
angelegt

Der Primärschlüssel in Access entspricht in SQL dem Anlegen einer Tabelle mit einer Spalte und dem Befehl `IDENTITY`.

Nachdem die Tabelle existiert, sollten Sie die einzelnen Datentypen der Felder einstellen. Dazu wechseln Sie in die Entwurfsansicht (`ANSICHT | ENTWURFSANSICHT`) und tragen hinter jedem Feldnamen (Abbildung 6.18) die entsprechenden Datentypen ein. Im unteren Teil des Dialogfeldes finden Sie weitere Parameter zu den einzelnen Datentypen. Um beispielsweise den Datentyp `INT` (in SQL) einzugeben, wählen Sie oben `ZAHN` und unten aus der Liste Feldgröße den Wert `INTEGER`.

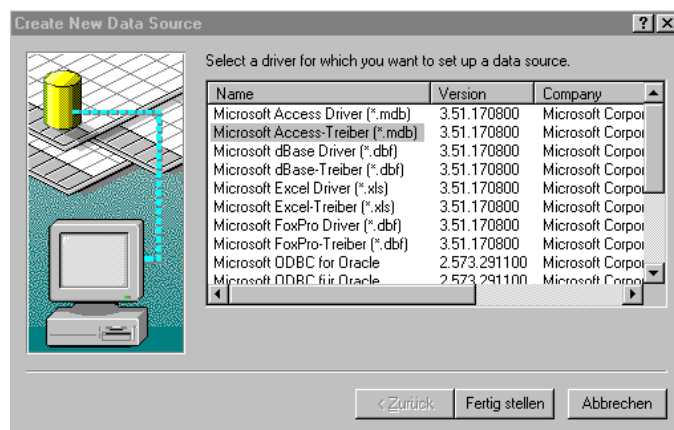
Abbildung 6.18:
Einstellungen der
Felddatentypen in
Access in der
Entwurfsansicht



Meldungen und Beschriftungen gelten nur innerhalb der Access-Umgebung und machen keinen Sinn, wenn Sie mit ASP auf die Access-Datenbank zugreifen.

Wenn Sie die Tabellen angelegt haben, muss die Access-Datenbank im System bekannt gemacht werden, damit die Datenobjekte darauf zugreifen können. Öffnen Sie dazu die `SYSTEMSTEUERUNG | ODBC`, wählen Sie die Registerkarte `SYSTEM DSN`.

Abbildung 6.19:
Auswahl des
Microsoft-Access-
Treibers beim
Anlegen einer
Datenquelle



Nach dem Klick auf FERTIG STELLEN wird die Quelle angelegt. Im folgenden Dialogfeld muss noch der Pfad zur Datenbank ausgewählt werden. Wählen Sie die von Access angelegte Datenbank (mit der Erweiterung .MDB). Die Datenquelle steht Ihnen nun zur Verfügung.

6.6.4 MySQL und ODBC

Da MySQL den Schwerpunkt der Datenbanknutzung in diesem Buch darstellt, darf ein entsprechender Ausflug in die ODBC-Welt von MySQL nicht fehlen. Dem universellen Charakter der PHP-Skripte in Bezug auf die Plattformunabhängigkeit steht natürlich die Nutzung einer »Windows-Only«-Datenbank extrem entgegen.



Die richtige Datei

MySQL-ODBC wird mit zwei DLLs geliefert. MYODBC.DLL ist die Standarddatei mit Debuggerunterstützung. MYODBC2.DLL ist hingegen auf Geschwindigkeit hin optimiert. Zum Austausch kann einfach die eine Datei über die andere kopiert werden.

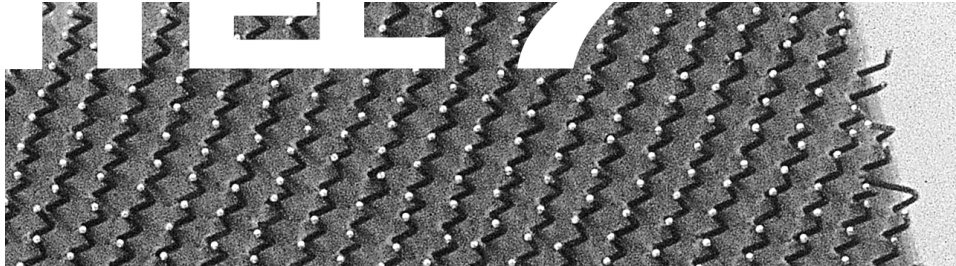
Nach der Installation, die keine Besonderheiten aufweist und automatisch durchläuft, können bei den ersten Versuchen Probleme auftreten:

Bekannte Probleme

- *Falscher Port eingestellt:* MySQL 3.20 verwendet Port 3333, während alle neueren Versionen Port 3306 verwenden.
- Häufig treten *Zugriffsfehler* auf. In diesem Fall sind die Zugriffsrechte in MySQL einzustellen. Dies hat nichts mit ODBC zu tun.
- *Falscher Treiber.* Der ODBC-Manager kann nicht feststellen, ob er unter Windows 9x/ME oder NT/2000 arbeitet. Es kann deshalb möglich sein, dass Sie die falsche Datei installieren. Der einzige Unterschied ist jedoch die Datei ODBC.INF.
- MyODBC 2.5 hat Probleme mit 64-Bit-Ganzzahlen.
- Umwandlungen von TIMESTAMP nach TIME arbeiten nicht richtig.
- Binärfelder werden als Zeichenkette zurückgegeben, nicht als hexadezimale Zeichenkette.
- Einige Datentypumwandlungen entsprechen nicht dem ODBC-Standard, das spielt aber nur selten eine Rolle.
- MyODBC hat keine volle Unterstützung für Datenbankcursor, obwohl der Treiber dies bei der Abfrage angibt.
- MyODBC unterstützt keine Nulldaten wie "0000-00-00". Solche Formen werden automatisch in NULL konvertiert.

6.6.5 ODBC-Funktionen

Um mit PHP auf ODBC zugreifen zu können, werden entsprechende Funktionen benötigt. Eine ausführliche Beschreibung der Funktionen mit Beispielen finden Sie am Ende von Kapitel 7 und im Referenzhandbuch zu PHP 4, das ebenfalls bei Carl Hanser unter dem Titel »PHP 4. Die Referenz« erschienen ist.



Datenbankprogrammierung

Der Umgang mit Datenbanken ist für den PHP-Programmierer besonders wichtig. Die Einbindung der Datenbankfunktionen ist einfach. Trotzdem sind Grundkenntnisse in der Abfragesprache SQL notwendig. Diese Kenntnisse werden anhand des Datenbankmanagementsystems MySQL diskutiert. Für die Anwender unter Windows wird das Zusammenspiel mit Access gezeigt.

MySQL-Funktionen (Seite 531)

Umgang mit Datenbanken (Seite 545)

Ein Datenbankprojekt entsteht (Seite 552)

Zugriff auf ODBC (Seite 570)

MySQL über Access bedienen (Seite 582)

Administration mit phpMyAdmin (Seite 586)

7.1 MySQL-Funktionen

MySQL wird von PHP explizit unterstützt. Für die Nutzung stehen Funktionen zur Verfügung, die direkt auf die Datenbank zugreifen. Das ermöglicht performante Skripte und eine flexible Programmierung. Andere Datenbanken werden in ähnlicher Weise angesprochen. Speziell für die Windows-Welt finden Sie die ➡ Abschnitte 7.4 *Zugriff auf ODBC* ab Seite 570 und ➡ 7.5 *MySQL über Access bedienen* ab Seite 582.

7.1.1 Funktionsübersicht

Die folgende Übersicht zeigt zur Orientierung alle Funktionen, mit denen MySQL angesprochen werden kann. **Alle Funktionen auf einen Blick**

- `mysql_affected_rows`
Gibt die Anzahl der von der letzten Operation betroffenen Zeilen zurück.
- `mysql_close`
Schließt eine Verbindung.
- `mysql_connect`
Öffnet eine Verbindung zum Server.
- `mysql_create_db`
Erzeugt eine Datenbank.
- `mysql_data_seek`
Setzt den internen Zeiger auf Ergebnisdatensätze.
- `mysql_db_query`
Sendet eine SQL-Abfrage an die Datenbank.
- `mysql_drop_db`
Löscht eine Datenbank.
- `mysql_errno`
Gibt die Fehlernummer der letzten Datenbankoperation zurück.
- `mysql_error`
Gibt die Fehlertexte der letzten Datenbankoperation zurück.

- `mysql_fetch_array`
Überträgt den aktuellen Ergebnisdatensatz in ein Array mit oder ohne Feldindizes.
- `mysql_fetch_assoc`
Überträgt den aktuellen Ergebnisdatensatz in ein Array mit Feldindizes.
- `mysql_fetch_field`
Holt Spalteninformationen eines Ergebnisdatensatzes und übergibt sie als Objekt.
- `mysql_fetch_lengths`
Ergibt die Länge der Ausgabe eines Ergebnisses.
- `mysql_fetch_object`
Überträgt den aktuellen Ergebnisdatensatz in ein Objekt.
- `mysql_fetch_row`
Überträgt den aktuellen Ergebnisdatensatz in ein nummeriertes Array.
- `mysql_field_name`
Gibt den Namen eines Felds im aktuellen Ergebnisdatensatz zurück.
- `mysql_field_seek`
Setzt den Ergebniszeiger auf ein spezifisches Feld.
- `mysql_field_table`
Gibt den Namen einer Tabelle zurück, die ein bestimmtes Feld enthält.
- `mysql_field_type`
Gibt den Typ eines Felds im Ergebnisdatensatz zurück.
- `mysql_field_flags`
Gibt die Parameter eines Felds im Ergebnisdatensatz zurück.
- `mysql_field_len`
Gibt die Länge eines Felds zurück.
- `mysql_free_result`
Löscht den von der Ergebnisliste benötigten Speicher und gibt ihn frei.

- `mysql_insert_id`
Gibt die ID des Autoinkrement-Felds eines vorangegangenen INSERT-Befehls zurück.
- `mysql_list_fields`
Mit dieser Funktion können alle Ergebnisfelder ausgegeben werden.
- `mysql_list_dbs`
Diese Funktion zeigt alle Datenbanken an.
- `mysql_list_tables`
Zeigt alle Tabellen einer Datenbank an.
- `mysql_num_fields`
Anzahl der Felder einer Ergebnisliste.
- `mysql_num_rows`
Anzahl der Datensätze (Reihen) der Ergebnisliste.
- `mysql_pconnect`
Öffnet eine persistente (ständige) Verbindung zu MySQL.
- `mysql_query`
Sendet eine SQL-Abfrage.
- `mysql_result`
Liest die Ergebnisliste der SQL-Anfrage ein.
- `mysql_select_db`
Diese Funktion wählt die Standarddatenbank aus.
- `mysql_tablename`
Ermittelt den Namen einer Tabelle zu einem gegebenen Feldnamen.

7.1.2 Erste Schritte mit MySQL und PHP

Für Anfänger sind Datenbanken immer etwas unheimlich. Die Anwendung ist zwar prinzipiell sehr einfach, dennoch gibt es einen wesentlichen Unterschied zur »normalen« PHP-Programmierung. Die Datenbank liegt außerhalb der PHP-Umgebung. Dadurch erhöht sich die Anzahl der Fehlerquellen. Außer im eigenen Skript kann der Fehler auch in der Datenbank (Tabelle falsch angelegt oder SQL-Befehl fehlerhaft) und in der Verbindung zwischen beiden liegen. Sie sollten

daher unbedingt den Umgang mit der Datenbank selbst üben, wie in Kapitel 6 beschrieben.

Voraussetzungen



Testen Sie Ihren Wissensstand vorab wie folgt:

- Können Sie MySQL am Prompt starten?
- Können Sie eine Datenbank und eine Tabelle mit dem MySQL-Monitor anlegen?
- Haben Sie die in Kapitel 6 gezeigten SQL-Beispiele ausprobiert und verstanden?

Sie sollten außerdem wissen, auf welchem Server MySQL läuft und welchen Nutzernamen und Kennwort den Zugriff ermöglichen. Konnten Sie alle Fragen positiv beantworten, ist es Zeit, die ersten Skripte auszuprobieren.

Mit der Datenbank verbinden

Der erste Schritt besteht in der Eröffnung einer Verbindung. Das folgende Skript macht nichts mit der Datenbank und gibt eine einfache Meldung aus, wenn alles glatt geht. Erhalten Sie eine Fehlermeldung, müssen Sie zuerst die Ursachen erforschen und beseitigen. Es ist sonst zwecklos, an dieser Stelle weiterzulesen. Im folgenden Skript ersetzen Sie *localhost* durch den Namen des Datenbankservers und *root* durch den Namen des Administrators, wenn dieser geändert wurde.

Listing 7.1:
mysql_connect: Mit
MySQL verbinden

```
<?php
$db = @mysql_connect("localhost","root");
if ($db) {
    echo "<p><b>Verbindung erfolgreich!</b> Handle-ID: $db";
} else {
    echo "<p><b>Fehler, keine Datenbankverbindung!</b>";
}
@mysql_close($db);
?>
```

Das Ergebnis sollte etwa wie in Abbildung 7.1 gezeigt aussehen.

Abbildung 7.1:
Erfolgreich mit
MySQL verbunden!

Die folgende Meldung informiert über den Erfolg der Aktion:
Verbindung erfolgreich! Handle-ID: Resource id #3

Fehlersuche

Erhalten Sie an dieser Stelle einen Fehler, sollten Sie zuerst auf Fehlersuche gehen:

- Ist der MySQL-Dienst oder -Daemon gestartet?
- Ist der Anmeldename korrekt?

- Wird eventuell ein Kennwort benötigt?
- Ist der Servername korrekt?
- Ist der Port verändert worden? Sie müssen dann den Port explizit angeben, an dem MySQL erreichbar ist.
- Haben Sie eine Verbindung zum MySQL-Server, wenn dies eine andere Maschine ist?

Hat es funktioniert, steht Ihnen das Datenbankhandle (im Beispiel in der Variabel *\$db*) zur Verfügung. Sie benötigen es, um bei allen anderen Funktionen jeweils den Bezug zur Verbindung herstellen zu können. So können zum einen mehrere Verbindungen parallel geöffnet werden. Zum anderen sparen Sie die wiederholte Angabe aller Parameter.

Die Funktion gibt FALSE zurück, wenn der Verbindungsversuch fehlschlug. Die Anwendung mit allen Parametern auf einen Blick: **mysql_connect()**

```
$handle = mysql_connect("server:port", "nutzer", "kennwort");
```

Die Funktion zum Schließen der Verbindung am Ende ist übrigens reine Schönheitskosmetik. PHP schließt normale Verbindungen am Ende des Skripts automatisch. Sie sollten dennoch immer alle Verbindungen explizit schließen, um Ressourcen sicher freizugeben. Außerdem ist die Programmierung sauberer.

7.1.3 Wichtige Funktionen

Der Zugriff auf die MySQL-Datenbank ist relativ einfach. Grundsätzlich kommen Sie mit den folgenden Funktionen aus:

- `mysql_connect` verbindet mit dem Datenbankserver.
- `mysql_select_db` wählt eine Datenbank aus.
- `mysql_query` führt eine SQL-Anweisung aus.
- `mysql_num_rows` ermittelt die Anzahl der Datensätze.
- `mysql_fetch_array` bzw. `mysql_fetch_assoc` holen die Daten in ein Array.

Komplexe Datenbankobjekte wie die ActiveX Data Objects (ADO), die u.a. in ASP verwendet werden, kennt PHP nicht. Sie müssen also alle Aufgaben direkt in SQL erledigen (deshalb auch die umfassende Darstellung in Kapitel 6). Der Vorteil ist eine deutlich höhere Leistung.



Hinweis



Die Übungsdatenbank anlegen

Um die folgenden Beispiele ausführen zu können, legen Sie eine kleine Übungsdatenbank an. Damit es nicht zu kompliziert wird, hilft Ihnen das Skript OPEN.INC.PHP. Dazu gehen Sie folgendermaßen vor:

- Führen Sie die Schritte nur aus, wenn Sie den Verbindungstest im letzten Abschnitt erfolgreich absolviert haben.
- Öffnen Sie das Skript OPEN.INC.PHP im Editor. Dieses Skript definiert einige Variablen und sorgt bei jedem Beispiel für die Verbindung zur Datenbank.
- Tragen Sie in die Variablen *\$server*, *\$user*, *\$pass* den Namen Ihres Datenbankservers, den Nutzernamen (Standard ist »root«) und das Kennwort ein. Wenn MySQL nicht am Standardport läuft, fügen Sie den Port mit Doppelpunkt dem Server hinzu, beispielsweise »localhost:3355«.
- Starten Sie das Skript MYSQL_CONNECT.PHP im Browser. Eine Ausgabe informiert Sie über den Erfolg der Aktion. Machen Sie erst weiter, wenn dieser Schritt erfolgreich war.

In jeder Beispieldatei dieses Abschnitts wird der Datenbankzugriff mit der folgenden Zeile aktiviert:

```
<?php include("../open.inc.php4"); ?>
```

In der Datei OPEN.INC.PHP4 finden Sie die schon bekannte `mysql_connect`-Funktion. Die im Folgenden abgedruckten Skripte sind auf die für das Verständnis wesentlichen Passagen gekürzt.

Die Datei selbst finden Sie im folgenden Listing:

*Listing 7.2:
open.inc öffnet die
Verbindung zur
Datenbank*

```
// Tragen Sie hier Ihre Werte ein!
$liveserver = ""; # Adresse des Liveservers ohne "www",
                # "www.meine.server.de = meine.server.de"
if ($SERVER_NAME == $liveserver) {
    # Werte auf Live-Server einstellen!
    $user = ""; # Username für die MySQL-DB
    $pass = ""; # Kennwort für die MySQL-DB
    $server = ""; # Adresse/IP/Name des MySQL-Server
    $dbase = ""; # Name der standardmäßig verwendeten Datenbank
} else {
    # Werte auf Entwicklungssystem einstellen!
    $server = "localhost"; // MySQL-Server
    $user = "root"; // MySQL-Nutzer
    $pass = ""; // MySQL-Kennwort
    $dbase = "customers"; // Standarddatenbank
}
//
$conn = @mysql_connect($server, $user, $pass);
if($conn) {
```

```
mysql_select_db($dbase, $conn);
} else {
    die("<b>Verbindung zum MySQL-Server konnte nicht hergestellt
        werden </b></body></html>");
} /* end if */
```

Das Skript erlaubt die Einstellung von zwei Werten für die erforderlichen Verbindungsparameter. Sie können dies verwenden, um sowohl lokal als auch auf dem Server Ihres Providers die selbst erstellten Skripte (oder die aus dem Buch) zu testen. Dazu ergänzen Sie am Anfang den Wert für *\$liveserver* mit der Domain, unter der Ihre Website läuft. Lokal dürfte fast immer »localhost« zutreffen. Binden Sie dann *open.inc.php4* in alle Skripte folgendermaßen ein:

```
include("open.inc.php4");
```

Wie es funktioniert

Datensätze lesen

Die Musterdatenbank enthält nach der ersten Installation 13 Datensätze. Eine Abfrage aller Datensätze erfolgt mit

mysql_query

```
$rs = mysql_query("SELECT * FROM tabelle", $conn)
```

Als Ergebnis wird ein Zeiger auf eine Ergebnisliste (engl. *result set*) zurückgegeben. Diesen Zeiger können Sie nutzen, um auf einzelne Datensätze zuzugreifen oder andere Funktionen anzusteuern. Die Anzahl der Datensätze können Sie mit folgendem Code ermitteln:

mysql_num_rows

```
$anzahl = mysql_num_rows($rs)
```

Das folgende Skript zeigt die Anwendung beider Funktionen:

```
<?php
$sql = "SELECT * FROM address";
$result = mysql_query($sql, $conn);
if ($result) {
    $number = mysql_num_rows($result);
    echo "<P>Es sind $number Datensätze gelesen worden.";
    ...
}
```

*Listing 7.3:
Ausschnitt aus
mysql_num_rows:
Abfrage der Anzahl
der Datensätze*

Als Nächstes möchten Sie die Datensätze sicher sehen. Die Abfrage der Ergebnisliste erfolgt in einer Schleife:

```
<?php
$sql = "SELECT * FROM address";
$result = mysql_query($sql, $conn);
if ($result) {
    $number = mysql_num_rows($result);
    echo "<p>Es sind $number Datensätze gelesen worden.";
    echo "<table border=1>";
    while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
        echo "<tr><td>";
        echo implode("</td><td>", $row);
        echo "</td></tr><n>";
    }
}
```

*Listing 7.4:
mysql_fetch_array:
Datensätze in einer
HTML-Tabelle
ausgeben*

```

    }
    echo "</table>";
    echo "<p>Abfrage: <i>$sql</i>";
} else {
    echo "<p>".mysql_error($conn);
}
mysql_close($conn);
?>

```

**mysql_fetch_array()
mysql_fetch_assoc()**

Diese Variante ist nun schon sehr viel leistungsfähiger, trotzdem sind es nur wenige Befehle, die eingesetzt werden müssen. Die Besonderheit steckt in der kleinen Schleife:

```
while ($row = mysql_fetch_array($result, MYSQL_ASSOC))
```

Die Funktion `mysql_fetch_array` überführt die Ergebnisliste in ein Array. Dabei werden zwei Parameter angegeben:

- `$result` ist der Zeiger auf die Ergebnisliste.
- `MYSQL_ASSOC` ist eine Konstante, welche die Art des Arrays bestimmt. Mehr dazu finden Sie in ➡ Abschnitt 7.2.3 *Die Ergebnisliste auslesen* ab Seite 546.

Alternativ können Sie ab PHP 4.0.4 `mysql_fetch_assoc` verwenden. Die Funktion entspricht `mysql_fetch_array` mit dem zusätzlichen Parameter `MYSQL_ASSOC`.

Der Rest der Schleife ist sehr einfach. Mit der Funktion `implode` wird das Array in eine Zeichenkette verwandelt, deren Trennzeichen gleich die Tabellenzellen bilden:

```

echo "<tr><td>";
echo implode("</td><td>", $row);
echo "</td></tr>";

```

Um ein Gefühl für die Arbeitsweise dieser sehr wichtigen Funktionen zu bekommen, sollten Sie auch das nächste Beispiel ausprobieren. Hier werden zusätzlich die Feldnamen ausgelesen und im Kopf der Tabelle angezeigt.

*Listing 7.5:
mysql_fetch_array2:
Auslesen der
Feldnamen zur
automatischen
Erstellung des
Tabellenkopfes*

```

<?php
$sql = "SELECT * FROM address";
$result = mysql_query($sql, $conn);
if ($result) {
    $number = mysql_num_rows($result);
    echo "<p>Es sind $number Datensätze gelesen worden.</p>";
    echo "<table border=1><tr>";
    $row = mysql_fetch_array($result, MYSQL_ASSOC);
    while ($field = key($row)) {
        echo '<th>' . ucwords($field) . '</th>';
        next($row);
    }
    echo "</tr><tr><td>";

```

```

echo implode("</td><td>", $row);
echo "</td></tr>";
while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
    echo "<tr><td>";
    echo implode("</td><td>", $row);
    echo "</td></tr>";
}
echo "</table>";
echo "<p>Abfrage: <i>${sql}</i>";
} else {
    echo "<p>".mysql_error($conn);
}
mysql_close($conn);
?>

```

Der Unterschied zum vorhergehenden Beispiel ist gering. Der Datensatz wird wie schon bekannt ermittelt: **Wie es funktioniert**

```
$row = mysql_fetch_array($result, MYSQL_ASSOC);
```

Die Schleife greift nun aber nicht direkt auf die Elemente zurück, sondern, da es sich um ein assoziatives Array handelt, auf die Indizes. Dazu wird die Funktion `key` verwendet. Mit `next` wird jeweils der interne Zeiger des Arrays verschoben. Das `<TH>`-Tag erzeugt eine Kopfzeile, bei der die Schrift automatisch fett erscheint:

```

while ($field = key($row)) {
    echo "<TH>${field}</TH>";
    next($row);
}

```

Der Rest des Skripts unterscheidet sich nur geringfügig. Die zusätzlichen `echo`-Befehle geben die erste Zeile der Tabelle aus, da sich der Datensatzzeiger mit jedem Auslesen eines Datensatzes eine Position weiter verschiebt. Die zusätzliche Ausgabe geht schneller als das Zurücksetzen des Zeigers auf den Anfang.

Es sind 13 Datensätze gelesen worden.

CustID	Fname	Sname	Company	Street	Zip	City	Telephone	Telefax	Email
1	Hopstein	Alexander	Conga Communications	Börselgasse	22765	Hamburg	040/1234356	040/9876541	hopstein@conga.de
2	Dannegger	Christian	Living-Systems	Roggenbachstrasse	78050	Villingen	07721/98910	07721/989191	dannegger@living-systems.de
3	Schatton	Peter	Euro Event	Tronjestrasse	44319	Dortmund	0231/212110	0231/21929	nomail
4	Borgmann	Georg	Euro Event	Tronjestrasse	44319	Dortmund	0231/212110	0231/21929	nomail
5	Jakob	Georg	Compuserve	Hauptstrasse	82008	Unterhaching	089/66571160	089/66571316	jakob@compuserve.com
6	Fernau	Peter	e-com AG	Richard-Wagner-Strasse	70184	Stuttgart	0711/24894812	0711/24894811	fernau@ecomag.de
7	Gaub	Sissi	e-com AG	Richard-Wagner-Strasse	70184	Stuttgart	0711/24894812	0711/24894811	info@ecomag.de
8	Kallola	Dan	e-com AG	Richard-Wagner-Strasse	70184	Stuttgart	0711/24894812	0711/24894811	info@ecomag.de
9	Eisenmenger	Frank	CIB	Robert-Rössle-Straße	13125	Berlin	030/94892911	030/94892912	cib@mdc-berlin.de
10	Meissner	Bernd	Openshop Holding AG	Wilhelmstrasse	89073	Ulm	0731/1553-100	0731/1552-111	info@openshop.de
11	Meene	Klaus	Living-Systems	Roggenbachstrasse	78050	Villingen	07721/98910	07721/989191	meene@living-systems.de
12	Neumeier	Frank	Ziff Davis Verlag	Riesstrasse	80992	München	089/14312142	089/14312100	neumeier@ziff.de
13	Nebel	Herbert		Mommsenstrasse	10629	Berlin	030/8814244		nebel@comvalley.com

Abfrage:
SELECT * FROM address

Abbildung 7.2:
Ausgabe der
Mustertabelle

Datensätze schreiben

Was aus der Datenbank herauskommt, muss auch irgendwie hineingekommen sein. Am Anfang hat da das kleine Installationsskript geholfen. Nun ist es an der Zeit, selbst ein paar Datensätze hinzuzufügen. Praktisch gibt es keine Unterschiede zum bereits gelernten. Nur die SQL-Anweisung heißt nun INSERT.

*Listing 7.6:
mysql_insert_id:
Anlegen eines neuen
Datensatzes*

```
<?php
$sql = "INSERT INTO address (fname, sname, email) ";
$sql .= "VALUES ('Krause', 'Joerg', 'joerg@krause.net')";
$result = mysql_query($sql, $conn);
if ($result) {
    $number = mysql_insert_id();
    echo "<P>Es wurde Datensatz Nr. $number erzeugt.";
} else {
    echo "<P>".mysql_error($conn);
}
mysql_close($conn);
?>
```

mysql_insert_id()

Vom Ergebnis können Sie sich anhand der angezeigten Datensatznummer und der Ausgabe des Skripts aus Listing 7.6 überzeugen. Neben der INSERT-Anweisung, die sicher keiner Erläuterung mehr bedarf, kam die Funktion `mysql_insert_id` hinzu. Diese Funktion bezieht sich immer auf die letzte Aktion an der Datenbank und benötigt keine Parameter:

```
$number = mysql_insert_id();
```

Andere Manipulationen

Die gezeigten Beispiele haben sicher verdeutlicht, dass die Datenbankfunktionen keinen nennenswerten Komfort bieten. Es wurde bereits angedeutet, dass sie sich dafür durch hohe Geschwindigkeit auszeichnen. Sie können praktisch jede beliebige SQL-Anweisung an die Datenbank senden und die Ergebnisse dort abholen. Der Rückgabewert der meisten Funktionen kann zur Ermittlung eines positiven Verlaufs herangezogen werden.

```
$rs = mysql_query(...);
```

Wenn ein Fehler auftrat oder keine Ergebnisse ermittelt werden konnten, ist der Wert 0, andernfalls steht in `$rs` ein internes Handle, mit dem auf die Ergebnisse zugegriffen werden kann. Für diesen Zugriff benötigen Sie immer entsprechende Funktionen wie beispielsweise `mysql_fetch_array`.

Daten löschen

Einzelne Daten löschen Sie mit der SQL-Anweisung DELETE. Die Umsetzung mit PHP unterscheidet sich nicht von den schon vorgestellten Skripten.

```
<?php
$fname = "Krause";
$sql = "DELETE FROM address WHERE ";
$sql .= "fname = '$fname'";
$result = mysql_query($sql, $conn);
if ($result) {
    $number = mysql_affected_rows($conn);
    echo "<P>Es wurden $number Datensätze gelöscht.";
} else {
    echo "<P>".mysql_error($conn);
}
mysql_close($conn);
?>
```

mysql_affected_rows()

Listing 7.7:
mysql_affected_rows:
Löschen von Datensätzen

Die Anzahl der gelöschten Datensätze hängt davon ab, wie oft Sie das vorangegangene Beispiel ausgeführt haben. Die Arbeitsweise selbst ist einfach. Die SQL-Anweisung wird direkt erzeugt. Der Suchwert, mit dem die zu löschenden Datensätze ausgewählt werden, wird mit einer Variablen erzeugt:

```
$fname = "Krause";
$sql = "DELETE FROM address WHERE ";
$sql .= "fname = '$fname'";
```

Wichtig ist die Angabe der einfachen Anführungszeichen, da MySQL das sonst nicht als Zeichenkette sondern als Spaltenbezeichner erkennt.

Neu ist die Anwendung der Funktion `mysql_affected_rows`, mit der die Anzahl der gelöschten Datensätze ermittelt wird:

```
$number = mysql_affected_rows($conn);
```

Die Anzahl der gelöschten Datensätze hängt davon ab, wieviele vorher mit dem Skript aus Listing 7.6 hinzugefügt wurden.

```
Es wurden 3 Datensätze gelöscht.
```

Abbildung 7.3:
Mögliche Reaktion des Skripts

Verwenden Sie das Installationsskript `INSTALL.SQL`, wenn Sie bei Manipulationen dieses Skripts versehentlich mehrere oder alle Datensätze der Beispieltabelle gelöscht haben.

Datenbanken anlegen und löschen

Das Löschen von Datenbanken und Tabellen ist ebenso einfach möglich. Für Datenbanken stehen sogar spezielle Funktionen zur Verfügung. Sie können trotzdem das Absenden von SQL-Anweisungen

nutzen. Das folgende Beispiel zeigt das Anlegen einer temporären Datenbank mit anschließender »Vernichtung«:

*Listing 7.8:
mysql create db:
Datenbanken anlegen
und löschen*

```
<?php
// Fragt Datenbanken ab
function show_dbs($conn) {
    $result = mysql_listdbs($conn);
    $i = 0;
    echo "<P><B>Datenbanken </B>:</P>";
    while ($i < mysql_num_rows ($result)) {
        $tb_names[$i] = mysql_tablename ($result, $i);
        echo $tb_names[$i] . "<BR>";
        $i++;
    }
}
// Schickt Abfrage an die Datenbank
function sql_query($success, $sql, $conn) {
    $result = mysql_query($sql, $conn);
    if ($result) {
        echo "<P>$success.";
    } else {
        echo "<P>".mysql_error($conn);
    }
    return $result;
}
$result = mysql_create_db("temporaer", $conn);
if ($result) {
    echo "<P>datenbank angelegt.";
} else {
    echo "<P>".mysql_error($conn);
}
show_dbs($conn);
$sql = "USE temporaer";
sql_query("Umschalten auf Temporaer", $sql, $conn);
$sql = "CREATE TABLE temp (sessions VARCHAR(50) NOT NULL)";
sql_query("Tabelle temp erzeugt", $sql, $conn);
$sql = "INSERT INTO temp (sessions) VALUES _
        ('67278492')";
sql_query("Session ID eingefügt", $sql, $conn);
$sql = "SELECT * FROM temp";
$result = sql_query("Session IDs anzeigen", $sql, $conn);
if ($result) {
    $number = mysql_num_rows($result);
    echo "<P>Es ist $number Datensatz gelesen worden: ";
    while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
        echo implode("|", $row);
    }
}
$result = mysql_drop_db("temporaer", $conn);
show_dbs($conn);
mysql_close($conn);
?>
```

Das Beispiel vermittelt alle Vorgänge, die zum Anlegen von Datenbanken und Tabellen mit PHP nötig sind. Zuerst wird eine Funktion definiert, die alle Datenbanken ermittelt:

mysql_listdbs()

```
$result = mysql_listdbs($conn);
```

Aus dieser Liste werden dann alle Tabellennamen extrahiert:

```
$i = 0;
echo "<P><B>Datenbanken </B>:</P>";
while ($i < mysql_num_rows($result)) {
    $tb_names[$i] = mysql_tablename($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
```

Mit der folgenden Anweisung legen Sie eine Datenbank an, in diesem Fall mit dem Namen *temporaer*:

mysql_create_db()

```
$result = mysql_create_db("temporaer", $conn);
```

Anschließend folgen einige SQL-Anweisungen, die Daten in die Datenbank einfügen. Der Ablauf kann als Standardanweisungsfolge genutzt werden:

- USE
Die neue Datenbank zur aktuellen erklären.
- CREATE TABLE
Tabellen anlegen.
- INSERT INTO
Daten einfügen.
- SELECT FROM
Daten wieder lesen.

Das explizite Löschen der Tabellen und Datensätze ist nicht notwendig, da die gesamte Datenbank anschließend vernichtet wird:

mysql_drop_db()

```
$result = mysql_drop_db("temporaer", $conn);
```

Der Vollständigkeit halber finden Sie an dieser Stelle die Möglichkeiten zum Löschen per SQL (siehe auch Kapitel 6 dazu):

- DROP TABLE löscht eine Tabelle.
- DROP DATABASE löscht die ganze Datenbank.
- DELETE FROM (ohne WHERE-Bedingung) löscht alle Datensätze der Datenbank.

- TRUNCATE TABLE löscht ebenfalls den gesamten Inhalt der Tabelle und ist schneller als DELETE.



Hinweis

Wenn Sie MySQL auf einem Server eines Providers nutzen, werden die Befehle für das Anlegen und Löschen von Datenbanken oft aus Sicherheitsgründen nicht zur Verfügung gestellt. Statt einer temporären Datenbank arbeiten Sie dann mit temporären Tabellen in Ihrer Stammdatenbank.

7.1.4 Persistente Verbindungen

Es gibt neben der normalen Verbindung zur Datenbank, die mit jedem Zugriff erneut geöffnet wird, auch eine schnellere Version. Diese so genannten persistenten Verbindungen werden in diesem Abschnitt vorgestellt.

Einführung

Wenn Sie in den vorangegangenen Beispielen die Skripte im Browser starteten, werden Sie sicher die relativ lange Reaktionszeit bemerkt haben. Die Ausgabe der Daten oder das Anlegen und Löschen der Tabellen selbst wird ausgesprochen schnell absolviert. Auch auf sehr schnellen Computern ist aber eine geringfügige Startverzögerung zu spüren. Das bemerkt natürlich auch der Nutzer.

Die Ursache liegt im Aufbau der Verbindung zur Datenbank. Nur hier entsteht die Verzögerung. Dies ist aber fatal, denn normalerweise werden die Verbindungen am Ende des Skripts wieder gelöscht. Jeder Zugriff auf einer der Folgeseiten erfährt wieder dieselbe Verzögerung.

PHP hat deshalb eine Unterstützung für persistente Verbindungen. Diese bleiben so lange bestehen, bis sie explizit wieder geschlossen werden oder eine einstellbare Zeit überschritten wurde.

Einschränkungen

Das folgende Beispiel und persistente Verbindungen im Allgemeinen können Sie nicht einsetzen (jedenfalls hat es keinen Effekt), wenn PHP als CGI läuft. Sie müssen PHP 3 als Modul im Apache laufen haben oder PHP 4 mit den Apache- oder ISAPI-Erweiterungen. Die Ursache wurde bereits bei den Sessions erläutert. Um festzustellen, welcher Nutzer welches Skript geöffnet hatte, muss das Verbindungshandle zugeordnet werden. Das geht nur, wenn dieses im Speicher gehalten werden kann. Da PHP unter CGI bei jedem Skriptaufruf erneut als Prozess startet, funktionieren persistente Verbindungen hier nicht. Sie können die Funktion `mysql_pconnect` zwar einsetzen, sie verhält sich aber ebenso wie `mysql_connect`.

Eine persistente Verbindung öffnen und schließen

Wann immer es die Programmlogik erlaubt, sollten persistente Verbindungen genutzt werden. Die entsprechende Funktion sieht so aus:

`mysql_pconnect()`
`mysql_close()`

```
$handle = mysql_pconnect("server:port", "nutzer", "kennwort");
```

In der Anwendung gibt es keine Besonderheiten. Ein Schließen der Verbindung mit `mysql_close` ist nicht möglich.

7.2 Umgang mit Datenbanken

Zur effektiven Programmierung von Datenbankfunktionen sind Zugriffe auf Datenbankcursor und -eigenschaften notwendig. Dieser Abschnitt zeigt die Vorgehensweise.

7.2.1 Datenbankcursor

In den vorangegangenen Abschnitten wurden die Cursor schon andeutungsweise genutzt. Hier geht es nun um diese Eigenschaft im Detail.



Aus Gründen der Effizienz werden nicht generell alle Daten an PHP übergeben. In den Beispielen in ➡ Abschnitt 7.1 *MySQL-Funktionen* ab Seite 531 standen nur wenige Daten zur Verfügung. Wenn Ihre Datenbank aber wächst und vielleicht 100 000 Datensätze enthält, die zudem jeweils einige KByte an Daten beinhalten, fällt es PHP sehr schwer, damit umzugehen. Praktisch müssten dann riesige Dateimengen in Variablen gehalten werden, was wiederum den Zugriff extrem verlangsamt. Die Belastung ist zwar auch bei der Verarbeitung im Kernel erheblich, dies wird jedoch nicht so schnell zum Engpass.

Einen Ausweg bieten die Datenbankcursor. Dies sind logische Zeiger, die auf einen bestimmten Datensatz der Ergebnisliste verweisen. Erst wenn Sie einen bestimmten Datensatz benötigen, beispielsweise um ihn zur Anzeige zu bringen, wird dieser (eine) Datensatz geholt. Zwangsläufig werden nun zusätzliche Funktionen benötigt, die mit solchen Zeigern umgehen können. Die Zeiger werden auch von einigen Funktionen beeinflusst, die keinen offensichtlichen Zugriff darauf haben. Oft wird der Zeiger beim Lesen eines Datensatzes weitergesetzt. Dies kann man sich in Schleifen zunutze machen. Andererseits müssen Sie genau wissen, welche Funktionen über solche Eigenschaften verfügen. Denn ohne diesen Transport der Zeiger würden die Schleifen endlos laufen.

Datenbankcursor

7.2.2 Eigenschaften der Ergebnisliste

Die Abfrage einer Datenbank gibt, wenn es sich um eine SELECT-Anweisung handelt, einen Zeiger auf eine Ergebnisliste zurück:

```
$rs = mysql_query("SELECT * FROM address");
```

mysql_num_rows()
mysql_num_fields() Dieser Zeiger kann nun zu verschiedenen Operationen herangezogen werden. Allgemeine Informationen über die Ergebnisliste erhalten Sie mit folgenden Funktionen:

- `mysql_num_rows(recordset)`

Ermittelt die Anzahl der Datensätze in der Ergebnisliste *recordset*.

- `mysql_num_fields(recordset)`

Die Funktion ermittelt die Anzahl der Spalten (Felder) der Ergebnisliste *recordset*.

Spezielle Funktionen für die Cursormanipulation

mysql_data_seek() Sie können den Zeiger gezielt mit der Funktion `mysql_data_seek` auf einen bestimmten Datensatz setzen:

```
$ok = mysql_data_seek($rs, $number);
```

Dabei bestimmt *\$number* den Datensatz innerhalb der Ergebnisliste. Die Funktion gibt 1 (True) zurück, wenn der Vorgang erfolgreich war, ansonsten 0 (FALSE).



Hinweis

Suchen Sie nicht nach einer Funktion, die »Nummer« des aktuellen Datensatzes zu ermitteln. SQL sortiert Datensätze nicht wie in einer Tabelle. Sie können also nicht genau vorhersagen, ob der Datensatz Nr. 14 der Ergebnisliste auch der Nummer 14 der Tabelle entspricht. Die Auswahl bestimmter Datensätze kann nur mit der WHERE-Bedingung vorgenommen werden.

7.2.3 Die Ergebnisliste auslesen

mysql_fetch_array()
mysql_fetch_assoc()
mysql_fetch_row()
mysql_fetch_object()
mysql_result() Um an die Werte in der Ergebnisliste zu kommen, stehen Ihnen folgende Funktionen zur Verfügung:

- `mysql_fetch_array`
- `mysql_fetch_assoc`
- `mysql_fetch_row`
- `mysql_fetch_object`
- `mysql_result`

In ➡ Abschnitt *Datensätze lesen* ab Seite 537 wurde bereits die Funktion `mysql_fetch_array` verwendet. Damit lesen Sie die aktuelle Ergebnisliste in ein Array ein. Die Funktion hat zwei Parameter:

```
$array = mysql_fetch_array($rs, ARRAYTYPE)
```

- `$rs` sollte das Resultat einer SQL-Abfrage enthalten (also ein Ergebnishandle wie es die Funktion `mysql_query` erzeugt).
- Der Parameter `ARRAYTYPE` ist optional, der Standardwert ist `MYSQL_BOTH`. Die Bedeutung der Konstanten im Einzelnen:

- `MYSQL_ASSOC`

Es wird ein assoziatives Array gebildet, das als Indizes die Feldnamen enthält und als Wert den Feldinhalt.

- `MYSQL_NUM`

Es wird ein numerisches Array erzeugt, die Felder werden von 0 beginnend durchnummeriert.

- `MYSQL_BOTH`

Es wird ein assoziatives Array erzeugt, dessen Indizes sowohl numerische Werte als auch Feldnamen enthalten. Dies ist die Standardeinstellung.

Wenn Sie den Parameter `MYSQL_ASSOC` vergessen und nicht nach numerischen Indizes selektieren, werden alle Werte doppelt ausgegeben, da sowohl die numerischen Indizes als auch die Schlüsselwerte auf basis der Feldnamen existieren.



Hinweis

Alternativ steht ab PHP 4.0.4 `mysql_fetch_assoc` zur Verfügung. Diese Funktion entspricht `mysql_fetch_array` mit dem Parameter `MYSQL_ASSOC` und ist geringfügig schneller.

Ähnlich ist die Funktion `mysql_fetch_row`. Die Funktion erzeugt immer ein indiziertes (numerisches) Array. Oft genügt dies, die Anwendung ist ausgesprochen einfach:

```
$array = mysql_fetch_row($rs);
```

Ein Geschwindigkeitsvorteil gegenüber `mysql_fetch_array` ergibt sich bei dieser Funktion allerdings nicht.

Wenn Sie viel mit Klassen und Objekten arbeiten, könnte die Funktion `mysql_fetch_object` von Vorteil sein:

```
$object = mysql_fetch_object($rs);
```

Die Felder werden hier als Eigenschaften abgebildet, der Zugriff sieht dann folgendermaßen aus:

```
$wert = $object->feldname;
```

Vor allem beim direkten Zugriff auf einzelne Felder mit bekannten Namen ist diese Version einfacher als die Anwendung assoziativer Arrays. Für eine umfassende Weiterverarbeitung der Daten sind Arrays wiederum eher zu empfehlen.

Wenn Sie nur ein bestimmtes Feld eines Datensatzes benötigen, bietet sich `mysql_result` an:

```
$wert = mysql_result($rs, $row, $field);
```

Dabei setzen Sie für *\$row* die Nummer der Reihe ein (analog zu `mysql_data_seek`) und für *\$field* den Index des Feldes (mit 0 beginnend) oder der Feldname. Als Feldname kann auch die Kombination »tabelle.feld« angegeben werden, wenn die Ergebnisliste mehrdeutige Namen für Spaltenbezeichner enthält.



Warnung

Die Funktion `mysql_result` ist langsamer und sollte nur in Ausnahmefällen Anwendung finden.

7.2.4 Informationen über die Datenbank

Wenn Sie universell programmieren wollen – was immer zu empfehlen ist –, erleichtern dynamische Informationen über die Datenbank die Arbeit.

Feldinformationen

`mysql_fetch_fields()`
`mysql_fetch_length()`

Beim Umgang mit Datenbanken kann es notwendig sein, Informationen über Datenbanken und Tabellen zu erhalten. Ihnen stehen für Informationen über die Felder der Ergebnisliste folgende Funktionen zur Verfügung:

- `mysql_fetch_fields`
- `mysql_fetch_length`

`mysql_fetch_fields` gibt ein Objekt zurück, dessen Eigenschaften die des Feldes repräsentieren:

```
$fieldobject = mysql_fetch_fields($rs, [$offset])
```

\$rs ist dabei wieder das Handle der Ergebnisliste, der optionale Parameter *offset* bestimmt das Feld innerhalb des aktuellen Datensatzes, über das Informationen geholt werden.

Die Eigenschaften des Objekts sind:

- `$fieldobject->name`
Der Spaltenname.
- `$fieldobject->table`
Der Name der zugehörigen Tabelle.
- `$fieldobject->max_length`
Die maximale Länge der Spalte.

- `$fieldobject->not_null`

Diese Eigenschaft gibt 1 (TRUE) zurück, wenn NULL nicht erlaubt ist.

- `$fieldobject->primary_key`

Diese Eigenschaft gibt 1 (TRUE) zurück, wenn das Feld einen Primärschlüssel hat.

- `unique_key`

Gibt 1 (TRUE) zurück, wenn das Feld UNIQUE (eindeutig) ist.

- `$fieldobject->multiple_key`

Diese Eigenschaft gibt 1 (TRUE) zurück, wenn das Feld nicht UNIQUE ist.

- `$fieldobject->numeric`

Gibt 1 (TRUE) zurück, wenn das Feld einen numerischen Datentyp hat.

- `$fieldobject->blob`

Diese Eigenschaft gibt 1 (TRUE) zurück, wenn das Feld ein BLOB-Feld ist.

- `$fieldobject->type`

Gibt den Datentyp zurück. Der Rückgabewert ist eine Zeichenkette mit dem Namen, also beispielsweise »blob« oder »int«.

- `$fieldobject->unsigned`

Gibt 1 (TRUE) zurück, wenn das Feld keine Vorzeichen zulässt.

- `$fieldobject->zerofill`

Diese Eigenschaft gibt 1 (TRUE) zurück, wenn das Feld standardmäßig 0 enthält.

Beachten Sie, dass hier die definierten Eigenschaften des Feldes ermittelt werden und nicht der tatsächliche oder vermeintliche Zustand. Das heißt, `$fieldobject->unsigned` ist 1, wenn das Feld negative Zahlen zulässt; es spielt keine Rolle, ob auch tatsächlich eine negative Zahl darin ist.

In diesem Zusammenhang ist auch `mysql_field_seek` interessant. Sie können mit dieser Funktion den internen Feldzeiger auf ein bestimmtes Feld setzen. Wenn dann `mysql_fetch_fields` ohne den optionalen Parameter *offset* verwendet, gilt die mit `mysql_field_seek` ausgewählte Spalte:

```
$ok = mysql_field_seek($rs, $offset);
```

Die Funktion gibt 0 (FALSE) zurück, falls der *\$offset* außerhalb des Wertebereichs liegt. Sie können auch Feldnamen angeben.

**mysql_fetch_
length()**

Die Funktion `mysql_fetch_length` gibt ein Array zurück, das Elemente für jede Spalte der Ergebnisliste enthält. Jedes Element repräsentiert dabei die Länge der Spalte in Byte. Damit lässt sich der Platzverbrauch oder die Nutzung von Variablen berechnen. Die Funktion kann nur angewendet werden, wenn zuvor auch eine Ergebnisliste erstellt und mit `mysql_fetch_row` ausgelesen wurde.

**mysql_field_type()
mysql_field_name()
mysql_field_len()
mysql_field_flag()**

Die Funktion `mysql_field_type` gibt nur den Datentyp eines Feldes zurück, `mysql_field_name` den Namen, `mysql_field_len` die Spaltenbreite (Anzahl der für die Ausgabe maximal benötigten Zeichen) und `mysql_field_flag` zusätzliche Eigenschaften.

*Listing 7.9:
mysql_fieldinfo zeigt
die Eigenschaften von
Spalten an*

```
<?php
// Datenbank muss verbunden sein. Passen Sie die Tabelle ggf. an!
$result = mysql_query("SELECT * FROM address");
$fields = mysql_num_fields($result);
$rows   = mysql_num_rows($result);
$i = 0;

$table = mysql_field_table($result, $i);
echo "<P>Die Tabelle ' $table ' hat <b>$fields</b> Felder und
      <b>$rows</b> Datensätze <BR>";

// Anzeige der Eigenschaften
echo "Die Tabelle hat folgende Felder:<BR>";
while ($i < $fields) {
    $type = mysql_field_type($result, $i);
    $name = mysql_field_name($result, $i);
    $len  = mysql_field_len($result, $i);
    $flags = mysql_field_flags($result, $i);
    echo "[".$type."] ".$name." (".$len.") ".$flags."<BR>";
    $i++;
}
mysql_close($conn);
?>
```

Wie es funktioniert

Die Funktion `mysql_field_flags` gibt die zusätzlichen Eigenschaften als Zeichenkette zurück, wobei einzelne Flags mit Leerzeichen getrennt sind. Sie können die Liste mit `explode` in ein Array umwandeln und so gezielt einzelne Werte extrahieren:

```
$flags = mysql_field_flags($rs, $offset);
$flag = explode(" ", $flags);
```

```
Die Tabelle 'address' hat 10 Felder und 13 Datensätze
Die Tabelle hat folgende Felder:
[int] custID (21) not_null primary_key auto_increment
[string] fname (50)
[string] sname (50)
[string] company (80)
[string] street (80)
[string] zip (6)
[string] city (40)
[string] telephone (40)
[string] telefax (40)
[string] email (50)
```

Abbildung 7.4:
Ausgabe der
Feldinformationen

Die Flags selbst sind Zeichenketten laut folgender Tabelle:

Feld	Beschreibung
not_null	Spalte akzeptiert keine Nullen.
primary_key	Spalte ist ein Primärschlüssel.
unique_key	Spalte akzeptiert nur eindeutige Werte.
multiple_key	Spalte akzeptiert auch mehrdeutige Werte.
blob	Spalte ist ein BLOB-Typ.
unsigned	Spalte akzeptiert Zahlen mit Vorzeichen.
zerofill	Spalte ist standardmäßig 0.
binary	Spalte akzeptiert Binärtypen.
enum	Spalte ist ein Aufzählungstyp.
auto_increment	Spalte verfügt über die Auto-Inkrement-Funktion.
timestamp	Spalte enthält einen Zeitstempel.

Tabelle 7.1:
Eigenschaften von
Feldern

Tabellen- und Datenbankeigenschaften

Ähnliche Funktionen stehen auch für Tabellen und Datenbanken zur Verfügung. Die Anwendung ist sicherlich seltener, denn eigentlich weiß man als Programmierer, welche Datenbankstruktur verwendet wird. Für universelle Schnittstellen, wie sie die in ➡ Abschnitt 7.6 *Administration mit phpMyAdmin* ab Seite 586 vorgestellte Anwendung *PHPMyAdmin* verwendet, ist der Einsatz aber durchaus sinnvoll.

Das letzte Beispiel verwendete bereits die Funktion `mysql_field_table`. Zurückgegeben wird der Name der Tabelle zu einem Feld. **`mysql_field_table()`**

```
$i = 0;
$table = mysql_field_table($result, $i);
```

Dies kann notwendig sein, wenn Sie in einer SELECT-Anweisung mit Aliassen und mehreren Tabellen arbeiten und nicht mehr wissen, aus welcher Tabelle die Daten stammen.

mysql_list_tables() Ergänzend können Sie auch mit den Funktionen `mysql_list_tables` und `mysql_tablename` die Namen aller Tabellen einer Datenbank ermitteln. Das folgende Skript zeigt die Anwendung.

Listing 7.10:
mysql_tablenames:
Anzeige aller
Tabellen einer
Datenbank

```
<?php
$result = mysql_list_tables($dbase);
$i = 0;
echo "<P>Tabellen in <B>$dbase</B>:</P>";
while ($i < mysql_num_rows($result)) {
    $tb_names[$i] = mysql_tablename($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

mysql_list_dbs() Ebenso kann natürlich eine Liste aller Datenbanken erstellt werden, hier ist als Parameter die Verbindung anzugeben:

Listing 7.11:
mysql_listdbs gibt
alle Datenbanken des
Servers aus

```
<?php
$result = mysql_list_dbs($conn);
$i = 0;
echo "<P>Datenbanken auf <B>$server</B>:</P>";
while ($i < mysql_num_rows($result)) {
    $tb_names[$i] = mysql_tablename($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
mysql_close($conn);
?>
```



Lassen Sie sich im letzten Beispiel nicht davon irritieren, dass auch die Liste der Datenbanken mit `mysql_tablename` ausgegeben wird. Prinzipiell gibt die Funktion nur eine Namensliste wieder. Um was es sich dabei handelt, spielt keine Rolle.

Abbildung 7.5:
Liste der
Datenbanken

```
Datenbanken auf localhost:
adressen
berlinshop
cookbook
customers
mauerfotos
mysql
rscout
test
```

Die konkrete Ausgabe hängt von der Konfiguration Ihres MySQL-Servers ab.

7.3 Ein Datenbankprojekt entsteht

Das folgende Datenbankprojekt basiert auf der Anwendung *phpHoo*, die Sie gut für die eigene Homepage einsetzen können. Dies ist weit- aus eleganter und flexibler als die ewig langen und unaktuellen Link-

listen und FFA²³-Listen. Mehr Informationen für dieses Projekt finden Sie unter:

www.webreference.com/perl/xhoo/php1

Entwickelt wurde *phpHoo* von Jonathan Eisenzopf u.a. Vom Autor ist diese Version überarbeitet und übersetzt worden. Im ➡ Abschnitt 9.6.6 *Das XML-Projekt* ab Seite 711 finden Sie eine Erweiterung, die ein frei verfügbares Verzeichnis in diese Datenbank importiert.

7.3.1 Vorbereitung

Am Anfang eines Datenbankprojekts steht immer die Schwierigkeit, keine Daten zu haben. Die Abfrage einer Datenbank ist prinzipiell einfacher als die Erzeugung. Es ist sicher bequem, ein fertige Datenbank zu nutzen. Um den Umgang richtig zu lernen, sollten Sie einen anderen Weg gehen. Entwickeln Sie die erste Datenbank »from scratch«²⁴, also ohne Hilfsmittel. Mit dem so erworbenen Verständnis können Sie anschließend bei eigenen Projekten deutlich produktiver arbeiten.

MySQL vorbereiten

In diesem Abschnitt wird viel mit dem MySQL-Monitor gearbeitet. Alle Schritte werden hier ausführlich wiedergegeben, wobei Eingaben fett gekennzeichnet sind und die Ausgaben in normalem Stil.



Hinweis

Starten Sie als Erstes den MySQL-Monitor:

```
$ mysql -u root
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 13 to server version: 3.21.29

Type 'help' for help.

mysql>
```

Jetzt befinden Sie sich am Prompt des MySQL-Monitors. An dieser Stelle können Sie jeden SQL-Befehl, den MySQL verstehen kann, eingeben.

Nun wird die Datenbank erzeugt. Der Schritt ist am einfachsten, Sie müssen nur einen Namen vergeben:

Datenbank erzeugen

```
mysql> create database mysearch;
Query OK, 1 row affected (0.16 sec)
```

²³ FFA = free for all; langweilige und sinnlose Linksammlungen.

²⁴ from scratch (engl.) = mit nichts anfangen, von Grund auf; in der Praxis heißt das: Rechner einschalten, System-Prompt abwarten, loslegen.

Die Datenbank wird nun mit dem Namen »mysearch« angelegt. Nun müssen Sie MySQL mitteilen, dass Sie mit dieser Datenbank arbeiten möchten:

```
mysql> use mysearch;
Database changed
```

Alle folgenden Kommandos, beispielsweise das Anlegen der Tabellen, beziehen sich nun auf die neue Datenbank.

Tabellen anlegen

Wenn Sie ein neues Projekt beginnen, sollten Sie sich über die Datenbankstruktur natürlich vorher im Klaren sein. Hier geht es vor allem darum, den Ablauf zu erlernen, folgen Sie also einfach den Anweisungen:

```
mysql> create table Categories(
-> CatID bigint(21) NOT NULL auto_increment,
-> CatName varchar(50) NOT NULL,
-> CatParent bigint(21),
-> PRIMARY KEY(CatID),
-> UNIQUE (CatName)
-> );
Query OK, 0 rows affected (0.11 sec)
```

Die Tabelle speichert die Kategorien der im Verzeichnis gespeicherten Seiten. Durch einen Verweis auf sich selbst kann eine beliebig tiefe Hierarchie in einer Tabelle gehalten werden. Zur Übersicht die Tabellenstruktur auf einen Blick:

Tabelle 7.2:
Struktur der Tabelle
»Categories«

Spaltenname	Datentyp	Besonderheit
<i>CatID</i>	bigint	Primärschlüssel, darf nicht Null sein
<i>CatName</i>	varchar	50 Zeichen, darf nicht Null sein, muss einmalig sein
<i>CatParent</i>	bigint	keine, verweist auf <i>CatID</i>

Beim Erzeugen der Tabelle werden alle Eigenschaften der Spalten sofort mit angegeben:

- Spaltenname
- Datentyp und (in Klammern) Länge des Datentyps
- NOT NULL, wenn Null-Werte nicht erlaubt sind
- auto_increment für ein Schlüsselfeld, das seinen Wert automatisch erhöht
- PRIMARY KEY definiert den Primärschlüssel
- UNIQUE legt fest, dass jeder Name eindeutig sein muss

MySQL verarbeitet nicht wirklich die Anweisung NOT NULL, wenn es sich um ein Zeichenfeld handelt. Wenn Sie aber mit einer anderen Datenbank arbeiten, kann die Angabe sinnvoll sein. Hier erfolgt der Einsatz aus Kompatibilitätsgründen.



Die Beschränkungen der Felder sind natürlich nicht zwingend notwendig, erleichtern aber die Fehlersuche. Denn MySQL wird Verstöße gegen die Regeln mit Fehlermeldungen ahnden, anstatt alles anstandslos durchgehen zu lassen und damit undurchschaubaren Datensalat zu erlauben.

In jeder Kategorie können, wie bei einem Verzeichnis üblich, beliebig viele Links stehen. Als Nächstes wird also eine Tabelle für die Hyperlinks benötigt. Die Struktur sollte folgendermaßen aussehen:

Spaltenname	Datentyp	Beschreibung
<i>LinkID</i>	bigint	Primärschlüssel, darf nicht Null sein, das Feld ist außerdem auto_increment
<i>CatID</i>	bigint	Zugeordnete Kategorie, darf nicht Null sein
<i>Url</i>	varchar(255)	Hyperlink, darf nicht Null sein, muss einmalig sein
<i>LinkName</i>	varchar(255)	Name des Hyperlinks
<i>Description</i>	varchar(255)	Beschreibung des Hyperlinks
<i>SubmitName</i>	varchar(80)	Name des Nutzers, der den Link sendet
<i>SubmitEmail</i>	varchar(80)	E-Mail des Nutzers, der den Link angemeldet hat
<i>Approved</i>	tinyint	Ist TRUE (1), wenn der Eintrag überprüft wurde

Tabelle 7.3:
Struktur der Tabelle
»Links«

Entsprechend dieser Beschreibung ist das Anlegen der Tabelle nun kein Problem mehr:

```
mysql> CREATE TABLE Links(
-> LinkID bigint(21) NOT NULL auto_increment,
-> CatID bigint(21) NOT NULL,
-> URL varchar(255) NOT NULL,
-> LinkName varchar(255) NOT NULL,
-> Description varchar(255),
-> SubmitName varchar(80),
-> SubmitEmail varchar(80),
-> Approved tinyint,
-> PRIMARY KEY (LinkID),
-> UNIQUE(URL)
```

```
-> );
Query OK, 0 rows affected (0.03 sec)
```

Das soll für den Anfang genügen. Sie können die Tabellen jederzeit um zusätzliche Spalten erweitern oder neue Tabellen hinzufügen, wenn dies erforderlich sein sollte. Bereits eingegebene Daten gehen dabei nicht verloren.

Dateneingabe

Es wird nun Zeit, ein paar Kategorien einzugeben. Dazu wird die SQL-Anweisung INSERT verwendet:

```
mysql> insert into Categories (CatName) values ('Meine Shops');
Query OK, 1 row affected (0.08 sec)
mysql> insert into Categories (CatName) values ('Banken');
Query OK, 1 row affected (0.01 sec)
mysql> insert into Categories (CatName) values ('Musik');
Query OK, 1 row affected (0.01 sec)
mysql> insert into Categories (CatName) values ('Computer');
Query OK, 1 row affected (0.00 sec)
```

Jetzt gibt es in der Tabelle *Categories* bereits vier Kategorien:

- Meine Shops
- Banken
- Musik
- Computer



Sie können die zuvor eingetippte Zeile im MySQL-Monitor wieder zur Anzeige bringen, indem Sie Cursor-Hoch (Pfeiltaste) drücken.

Beachten Sie bei der Eingabe der Kategorien, dass *CatParent* nicht vergeben wird. MySQL trägt hier standardmäßig NULL ein, diese Kategorie bilden also Einträge der obersten Ebene.

Datenausgabe

Lassen Sie sich nun die Kategorien der ersten Ebene anzeigen:

```
mysql> select * from Categories;
+-----+-----+-----+
| CatID | CatName | CatParent |
+-----+-----+-----+
| 1     | Meine Shops | NULL |
| 2     | Banken      | NULL |
| 3     | Musik       | NULL |
| 4     | Computer    | NULL |
+-----+-----+-----+
4 rows in set (0.05 sec)
```

Sie sehen links in der Tabelle den automatisch erzeugten Wert für *CatID*. Wenn Sie Ihre Lieblingsbanken nun unterhalb der Kategorie »Banken« eintragen möchten, geben Sie Folgendes ein:

```
mysql> insert into Categories (CatName, CatParent) values
('Deutsche Bank',2);
Query OK, 1 row affected (0.01 sec)
mysql> insert into Categories (CatName, CatParent) values
('Berliner Bank',2);
Query OK, 1 row affected (0.00 sec)
mysql> insert into Categories (CatName, CatParent) values
('Consors',2);
Query OK, 1 row affected (0.00 sec)
```

Wählen Sie nun erneut die SELECT-Anweisung an:

```
mysql> select * from Categories;
```

Schauen Sie sich die Tabelle aufmerksam an und versuchen Sie, die Zuordnung mit Hilfe der Spalte *CatParent* zu verstehen.

CatID	CatName	CatParent
1	Meine Shops	NULL
2	Banken	NULL
3	Musik	NULL
4	Computer	NULL
5	Deutsche Bank	2
6	Berliner Bank	2
7	Consors	2

Um die Datensätze einer bestimmten Kategorie zur Anzeige zu bringen, geben Sie Folgendes ein:

```
mysql> select * from Categories where CatParent = 2;
```

CatID	CatName	CatParent
5	Deutsche Bank	2
6	Berliner Bank	2
7	Consors	2

So einfach ist das! Die Auswahl der Kategorien der obersten Ebene ist ein klein wenig schwieriger, denn ein Vergleich mit NULL ist nicht zulässig, statt dessen wird *CatParent* IS NULL geschrieben:

```
mysql> select * from Categories where CatParent IS NULL;
```

CatID	CatName	CatParent
1	Meine Shops	NULL
2	Banken	NULL
3	Musik	NULL
4	Computer	NULL

Nun sind alle Befehle vorhanden, um mit der Datenbank arbeiten zu können, auch einige Testdaten existieren. Jetzt ist es an der Zeit, die PHP-Skripte zu schreiben und für etwas mehr Komfort zu sorgen.

7.3.2 Die Stammfunktionen

Um eine effektive Applikation zu schreiben, sind einige Funktionen notwendig, die später ein hohes Maß an Flexibilität erlauben. Die Anwendung soll deshalb mit Objekten arbeiten.

MySQL-Funktionen Zunächst erfolgt die Definition einer Klasse mit dem naheliegenden Namen MySQL. Der erste Teil erzeugt einige globale Variablen, welche die Daten für den Datenbankzugriff enthalten, *\$DBASE* speichert den Namen der Datenbank, *\$USER* den Nutzernamen, *\$PASS* das Kennwort und *\$SERVER* den Servernamen (bei Ihnen wird hier vermutlich localhost stehen). Zur Konfiguration müssen nur die vier Variablen geändert werden, sonst nichts. Alles andere sind Hilfsvariablen, die wir später benötigen.

*Listing 7.12:
Das Skripts phpHoo
(in mehreren
kommentierten
Abschnitten)*

```
<?php
class MySQL {

    var $CONN = "";
    var $DBASE = "mysearch";
    var $USER = "root";
    var $PASS = "";
    var $SERVER = "localhost";

    var $TRAIL = array();
    var $HITS = array();
    var $AUTOAPPROVE = true;
```

Die folgende Funktion soll Fehler ausgeben:

↓

```
function error($text) {
    $no = mysql_errno();
    $msg = mysql_error();
    echo "[$text] ( $no : $msg )<BR>\n";
    exit;
}
```

Objekte einer Klasse müssen initialisiert werden, die folgende Methode *init* erledigt dies. Zuerst werden die globalen Variablen in lokale (innerhalb der Methode gültige) übertragen, dann wird die Verbindung geöffnet. Der Befehl *\$this->error* ruft die Fehlerfunktion auf. In der globalen Variablen *\$CONN* wird anschließend das Datenbank-handle gespeichert:

↓

```
function init () {
    $user = $this->USER;
    $pass = $this->PASS;
```

```

$server = $this->SERVER;
$dbase   = $this->DBASE;
$conn    = mysql_connect($server,$user,$pass);
if(!$conn) {
    $this->error("Keine Verbindung!");
}
if(!mysql_select_db($dbase,$conn)) {
    $this->error("Keine Datenbank!");
}
$this->CONN = $conn;
return true;
}

```

Da die Abfrage mit SELECT häufiger benötigt wird, bietet es sich an, dafür eine eigene Methode zu schreiben:

```
function select ($sql="") {
```

↓

Zuerst werden Fehler abgefangen. `empty($sql)` erkennt leere Befehlsübergaben, mit `eregi("^select",$sql)` wird ermittelt, ob der Befehl auch mit der Zeichenfolge *select* beginnt. Wenn Sie auch Großschreibung wünschen, ersetzen Sie `eregi` durch `ereg`.

```

if(empty($sql)) { return false; }
if(!eregi("^select",$sql))
{
    echo "<H2>Falscher Befehl!</H2>\n";
    return false;
}

```

↓

Bei fehlender Verbindung soll auch nichts passieren, dies vermeidet lästige Folgefehler. Besteht eine Verbindung, wird diese in eine lokale Variable übertragen:

```

if(empty($this->CONN)) { return false; }
$conn = $this->CONN;

```

↓

Dann wird die Abfrage direkt ausgeführt und das Ergebnis in ein Abfrageobjekt überführt:

```
$results = mysql_query($sql,$conn);
```

↓

War nichts zu holen, ist die Methode beendet:

```

if( (!$results) or (empty($results)) ) {
    mysql_free_result($results);
    return false;
}

```

↓

Sind Daten vorhanden, werden diese nun reihenweise in ein Array geholt:

```

$count = 0;
$data = array();
while ( $row = mysql_fetch_array($results) )

```

↓

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E


```

    {
        $data[$count] = $row;
        $count++;
    }

```

Zum Schluss wird der für die Ergebnisliste verwendete Speicher freigegeben und das Array zurückgegeben:

↓

```

    mysql_free_result($results);
    return $data;
}

```

Ähnlich funktioniert die Methode *insert*, mit der Daten hinzugefügt werden. Die Methode gibt die ID-Nummer des eingefügten Datensatzes zurück:

↓

```

function insert ($sql="") {
    if(empty($sql)) { return false; }
    if(!ereg("insert",$sql)) {
        echo "<H2>Falscher Befehl!</H2>\n";
        return false;
    }
    if(empty($this->CONN)) { return false; }
    $conn = $this->CONN;
    $results = mysql_query($sql,$conn);
    if(!$results) { return false; }
    $results = mysql_insert_id();
    return $results;
}

```

Funktionen für den Suchkatalog

Die folgenden Methoden übernehmen spezielle Aufgaben für den Suchkatalog. Hier eine Übersicht:

- *get_Cats*
- *get_ParentsInt*
- *get_CatIDFromName*
- *get_CatNames*
- *get_Links*
- *get_CatFromLink*
- *search*
- *suggest*

Die Methode *get_Cats* ermittelt alle Kategorien unterhalb einer angegebenen Kategorie. Damit wird später die Navigation per Mausklick von einer höheren zu einer niedrigeren Ebene realisiert. Wegen der schon am Anfang angesprochenen Problematik mit dem Vergleich gegen NULL erfolgt zuerst eine entsprechende Abfrage:

```
function get_Cats ($CatParent= "") {  
    if(empty($CatParent)) {  
        $CatParent = "IS NULL";  
    } else {  
        $CatParent = "= $CatParent";  
    }  
}
```



Dann wird eine SQL-Anweisung erstellt und mit Hilfe der Methode *select* ausgeführt:

```
$sql = "SELECT CatID,CatName FROM Categories  
        WHERE CatParent $CatParent  
        ORDER BY CatName";  
$results = $this->select($sql);  
return $results;  
}
```



Um den gesamten Kategorien-Baum durchlaufen zu können, wird die folgende Methode genutzt.

```
function get_ParentsInt($CatID="") {  
    if(empty($CatID)) { return false; }  
    unset($this->TRAIL);  
    $this->TRAIL = array();  
    $this->get_Parents($CatID);  
}
```



Die Methode *get_Parents* wird nur von *get_ParentsID* aufgerufen, nie direkt.

```
function get_Parents ($CatID="") {  
    if((empty($CatID)) or (" $CatID" == "NULL")) {  
        return false;  
    }  
    $sql = "SELECT CatID,CatParent,CatName  
            FROM Categories where CatID = $CatID";  
    $conn = $this->CONN;  
    $results = mysql_query($sql,$conn);  
    if( (!$results) or (empty($results)) ) {  
        mysql_free_result($results);  
        return false;  
    }  
    while ( $row = mysql_fetch_array($results)) {  
        $trail = $this->TRAIL;  
        $count = count($trail);  
        $trail[$count] = $row;  
        $this->TRAIL = $trail;  
        $id = $row["CatParent"];  
        $this->get_Parents($id);  
    }  
    return true;  
}
```



V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Wenn der Nutzer nach einem Namen sucht, muss die passende ID gefunden werden. *get_CatIDFromName* erledigt dies:

```
↓
function get_CatIDFromName($CatName="") {
    if(empty($CatName)) { return false; }
    $sql = "SELECT CatID from Categories
           WHERE CatName = '$CatName'";
    $results = $this->select($sql);
    if(!empty($results)) {
        $results = $results[0]["CatID"];
    }
    return $results;
}
```

Umgekehrt wird anhand der ID der Name ermittelt:

```
↓
function get_CatNames( $CatID="") {
    if($CatID == 0) { return "Top"; }
    $single = false;
    if(!empty($CatID)) {
        $single = true;
        $CatID = "WHERE CatID = $CatID";
    }
    $sql = "SELECT CatName from Categories $CatID";
    $results = $this->select($sql);
    if($single) {
        if(!empty($results)) {
            $results = $results[0]["CatName"];
        }
    }
    return $results;
}
```

Mit *get_Links* werden die Hyperlinks innerhalb einer Kategorie ermittelt:

```
↓
function get_Links($CatID = "") {
    if(empty($CatID)) {
        $CatID = "= 0";
    } else {
        $CatID = "= $CatID";
    }
    $sql = "SELECT Url,LinkName,Description
           FROM Links
           WHERE (Approved != 0)
           AND CatID $CatID ORDER BY Url";
    $results = $this->select($sql);
    return $results;
}
```

Wird nur ein Link eingegeben, kann umgekehrt auch die richtige Kategorie ermittelt werden:

```
function get_CatFromLink($LinkID="") {  
    if(empty($LinkID)) { return false; }  
    $sql = "SELECT CatID FROM Links  
        WHERE LinkID = $LinkID";  
    $results = $this->select($sql);  
    if(!empty($results)) {  
        $results = $results[0]["CatID"];  
    }  
    return $results;  
}
```

↓

Suchfunktionen bringen erst den richtigen Komfort für den Nutzer. In der Funktion *search* ist dies implementiert:

```
function search ($keywords = "") {  
    if(empty($keywords)) { return false; }  
    $keywords = trim(urldecode($keywords));
```

↓

Mehrere Leerzeichen werden zu einem zusammengefasst:

```
$keywords = ereg_replace("[ ]+", " ", $keywords);
```

↓

Sind mehrere Wörter vorhanden, werden diese in ein Array extrahiert:

```
if(!ereg(" ", $keywords)) {  
    // Nur ein Suchwort  
    $KeyWords[0] = "$keywords";  
} else {  
    $KeyWords = explode(" ", $keywords);  
}
```

↓

Jetzt wird die SQL-Anweisung vorbereitet:

```
$sql = "SELECT DISTINCT  
    LinkID, CatID, Url, LinkName, Description  
    FROM Links  
    WHERE (Approved != 0)  
    AND ( $DEBUG ";  
$count = count($KeyWords);
```

↓

Falls nur ein Wort angefragt wurde, wird die entsprechende WHERE-Bedingung mit LIKE fertiggestellt:

```
if( $count == 1) {  
    $single = $KeyWords[0];  
    $sql .= " (Description LIKE '%$single%'  
        OR (LinkName LIKE '%$single%'  
        OR (Url LIKE '%$single%') )";
```

↓

Ansonsten wird für jedes Wort eine Abfrage generiert und mit OR (oder) verknüpft:

```
} else {  
    $ticker = 0;  
    while (list ($key, $word) = each ($KeyWords)) {  
        $ticker++;
```

↓

```

        if(!empty($word)) {
            if($ticker != $count) {
                $sql .= " ( (Description LIKE '%$word%')
                        OR (LinkName LIKE '%$word%')
                        OR (Url LIKE '%$word%') ) ";
            } else {
                // letztes OR vermeiden
                $sql .= " ( (Description LIKE '%$word%')
                        OR (LinkName LIKE '%$word%')
                        OR (Url LIKE '%$word%') ) ";
            }
        }
    } /* end while */

```

Zum Schluss wird die SQL-Anweisung komplettiert:

↓

```

        $sql .= " ) ORDER BY LinkName $DEBUG";
    }
    if(!empty($DEBUG)) {
        echo "<PRE>$sql\nTicker [$ticker]\nCount $count]</PRE>\n";
    }
    $results = $this->select($sql);
    return $results;
}

```

Mit der Funktion *suggest* werden neue Vorschläge in die Suchmaschine eingetragen:

↓

```

function suggest ($postData="") {
    if( (empty($postData)) or (!is_array($postData)) ) {
        return false;
    }
    $CatID = $postData["CatID"];
    $Url = $postData["Url"];
    $Description = $postData["Description"];
    $LinkName = $postData["LinkName"];
    $SubmitName = $postData["SubmitName"];
    $SubmitEmail = $postData["SubmitEmail"];
    $SubmitDate = time();
}

```

Zuerst folgen diverse Fehlerabfragen: leere Felder werden nicht eingetragen. Hier können Sie weitere Prüfungen auf Sinnfälligkeit implementieren:

↓

```

    if(empty($Url)) { return false; }
    if(empty($Description)) { return false; }
    if(empty($LinkName)) { return false; }
    if(empty($SubmitName)) { return false; }
    if(empty($SubmitEmail)) { return false; }
}

```

Mit *\$Approved* kann der Eintrag bis zu einer Prüfung gesperrt werden. Standardmäßig ist *\$this->AUTOAPPROVE* auf TRUE eingestellt, Links erscheinen deshalb sofort.

```
$Approved = 0;
if($this->AUTOAPPROVE) { $Approved = 1; }
```

↓

Nun kann der Datensatz erzeugt werden:

```
$sql = "INSERT INTO Links ";
$sql .= "(CatID, Url, LinkName, Description, SubmitName,
        SubmitEmail, SubmitDate, Approved) ";
$sql .= "VALUES ";
$sql .= "($CatID, '$Url', '$LinkName', '$Description',
        '$SubmitName', '$SubmitEmail', $SubmitDate,
        $Approved) ";
$results = $this->insert($sql);
return $results;
}
} /* End Class */
?>
```

↓

Mit diesen Funktionen kann nun die eigentliche Umgebung programmiert werden.

7.3.3 Realisierung

Der nächste Schritt besteht in der Erstellung der Skripte, mit denen die eigentliche Nutzeroberfläche geschaffen wird. Dank der Klassen gestaltet sich dies relativ einfach.

Vorstellung im Detail

Zuerst wird die bereits vorgestellte Datei [search](#) eingebunden:

```
<?
include("../search.php");
```

Listing 7.13:
*search: Suchfunktion
in phpHoo*

Dann wird ein Objekt der Klasse MySQL erzeugt. Tritt ein Fehler auf, endet das Skript:

```
$db = new MySQL;
if(!$db->init()) {
    die("Ein Fehler ist aufgetreten<BR>\n");
}
```

↓

Es folgt eine Funktion *breadcrumbs*, die den Suchpfad durch das Verzeichnis erstellt und die Links zur Navigation generiert:

```
function breadcrumbs($CatID="") {
    global $db;
    global $PHP_SELF;

    if(empty($CatID)) { return; }
    $db->get ParentsInt($CatID);
    $path = $db->TRAIL;
    if(!empty($path)) {
```

↓

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

```

        while ( list ( $key,$val ) = each ($path)) {
            $CatID      = stripslashes($val["CatID"]);
            $CatName     = stripslashes($val["CatName"]);
            $trail       = "|<A HREF=\"$PHP_SELF?
                        viewCat=$CatID\"><B>$CatName</B></A>$trail";
        }
    } else {
        $trail = "";
    }
    return $trail;
}

```

Die Anzeige der Seite beginnt immer mit der Funktion *start_page*, die folgendermaßen implementiert ist:

↓

```

function start_page($CatID="", $title="", $msg="") {
    global $PHP_SELF;

```

Zunächst wird der Kopf der Seite erzeugt. Hier ist ein guter Platz, um ein eigenes Logo einzubauen:

↓

```

    echo <<<HEAD1
    <HTML>
    <HEAD><TITLE>phpHoo - $title</TITLE></HEAD>
    <BODY BGCOLOR="#FFFFFF" TEXT="#000000" LINK="#0033FF"
        VLINK="#660099">
    <CENTER><H1>phpHoo</H1></CENTER>
HEAD1;

```

Falls es bei einer vorherigen Aktion eine Nachricht gab, wird diese hier ausgegeben:

↓

```

    if(!empty($msg)) {
        echo "\n<CENTER><B>$msg</B></CENTER>\n";
    }

```

Dann wird das Suchformular erzeugt:

↓

```

    echo <<<HEAD2
    <CENTER><FORM ACTION="$PHP_SELF" METHOD="POST">
    <INPUT TYPE="TEXT" NAME="KeyWords" SIZE=20>
    <INPUT TYPE="SUBMIT" NAME="Search" VALUE="Search">
    </FORM></CENTER>
    <P><H3><A HREF="$PHP_SELF">Top</A>
HEAD2;

```

Zuletzt wird der Pfad ausgegeben, der zur Navigation im Verzeichnis dient:

↓

```

    $trail = breadcrumbs($CatID);
    echo "$trail</H3></P><HR>\n";
    return;
}

```

Die Funktion *start_browse* durchsucht nun alle Kategorien und zeigt die Links in der ausgewählten Kategorie an:

```
function start_browse($CatID = "") {
    global $PHP_SELF;
    global $db;
    $data    = $db->get_Cats($CatID);
    $links   = $db->get_Links($CatID);

    if(!empty($CatID)) {
        $currentID = $CatID;
        $currentName = $db->get_CatNames($CatID);
    }
```

↓

Falls keine Kategorie angegeben wurde, wird »Top« angenommen:

```
} else {
    $currentID = "top";
    $currentName = "top";
}
```

↓

Sind Kategorien auf der ausgewählten Ebene, werden diese als Liste angezeigt:

```
if(!empty($data)) {
    while ( list ( $key,$val ) = each ($data)) {
        $CatID = stripslashes($val["CatID"]);
        $CatName = stripslashes($val["CatName"]);
        print "<LI> <A HREF=\"\$PHP_SELF?
viewCat=$CatID\"><B>$CatName</B></A></LI>\n";
    }
}
```

↓

Eine Linie trennt Kategorien und Links:

```
echo "<HR>\n";
```

↓

Falls Links existieren, werden diese nun angezeigt:

```
if(is_array($links)) {
    while ( list ( $key,$val ) = each ($links)) {
        $Url      = stripslashes($val["Url"]);
        $LinkName = stripslashes($val["LinkName"]);
        $Desc     = stripslashes($val["Description"]);
        print "<LI>
<A HREF=\"\$Url\"><B>$LinkName</B></A> - $Desc</LI>\n";
    }
}
```

↓

Zum Schluss wird noch der Link erzeugt, der zum Formular für eigene Vorschläge führt:

```
echo "<P><CENTER><A HREF=\"\$PHP_SELF?add=$currentID\">
    Einen neuen Link vorschlagen</A></CENTER>\n";
echo "</BODY></HTML>\n";
```

↓

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E


```

    return;
}

```

Jetzt beginnt der Hauptteil des Skripts mit dem Aufruf der Funktionen *start_page* und *start_browse*:

↓

```

/*****
$query = getenv("QUERY_STRING");
if( ($viewCat) or ( ($HTTP_POST_VARS) and ($query) ) ) {
    start_page($viewCat);
    start_browse($viewCat);
    exit;
}

```

Falls der Link zum Vorschlagsformular ausgewählt wurde, enthält die Variable *\$add* den Wert »top«. Dann wird das Formular erzeugt:

↓

```

if($add) {
    if($add == "top") { $add = 0; }
    $CatName = stripslashes($db->get_CatNames($add));
    if(empty($CatName)) { $CatName = "Top"; }
    echo <<<SUGGEST
    <HTML>
    <head><title>phpHoo - Link hinzuf&uuml;gen</title></head>
    <body BGCOLOR="#FFFFFF" TEXT="#000000" LINK="#0033FF"
        VLINK="#660099">
    <center><H1>phpHoo</H1></center>
    <p><H3>Einen Link zu <I>$CatName</I> hinzuf&uuml;gen</H3>
    <HR noshade>
    <form action="$PHP_SELF" method="POST">
    <input type="hidden" name="CatID" value="$add">
    <table border="0" cellpadding="2" cellspacing="2">
    <tr><td align="right"><B>URL:</B></td>
    <td><input name="Url" size="40" VALUE="http://"></td></tr>
    <tr><td align="right"><B>Titel:</B></td>
    <td><input name="LinkName" size="40"></td></tr>
    <tr><td align="right"><B>Beschreibung:</B></td>
    <td><textarea name="Description" rows="3" cols="40">
        </textarea></td></tr>
    <tr><td align="right"><B>Ihr Name:</B></td>
    <td><input name="SubmitName" size="40"></td></tr>
    <tr><td align="right"><B>Ihre E-Mail:</B></td>
    <td><input name="SubmitEmail" size="40"></td></tr>
    <tr><td></td>
    <td><input type="submit" name="suggest"
        value="Vorschlag absenden">
    <input type="reset" value=" Leeren "></td></tr>
    </table>
    <HR noshade>
    </form></p>
    </html>
SUGGEST;

```

Falls das Formular abgesendet wurde, wird in Abhängigkeit von der Prüfung der Daten eine entsprechende Meldung erzeugt:

```
} elseif ($suggest) {
    $junk = "";
    if(!$db->suggest($HTTP_POST_VARS)) {
        $title = "Vorschlagsfehler";
        $msg = "Vorschlag misslungen! Daten fehlten
                oder der Eintrag existiert bereits.\n";
    } else {
        $title = "Vorschlag empfangen";
        $msg = "Der Vorschlag ist zur Prüfung
                eingegangen\n";
    }
}
```

↓

Dann wird die Seite mit den Meldungstexten erneut angezeigt:

```
start_page($junk,$title,$msg);
start_browse();
exit;
```

↓

Der nächste Block wird aufgerufen, wenn eine Suchanfrage erfolgte:

```
} elseif ($KeyWords) {
    //start_page();
    $hits = $db->search($KeyWords);
```

↓

Wurden keine Suchergebnisse gefunden, wird eine entsprechende Meldung erzeugt:

```
if( (!$hits) or (empty($hits)) ) {
    $junk = "";
    $title = "Suchergebnisse";
    $msg = "Keine Seiten gefunden";
    start_page($junk,$title,$msg);
```

↓

Andernfalls wird die Anzahl der Treffer angezeigt und danach eine Liste der Treffer:

```
} else {
    $total = count($hits);
    $title = "Suchergebnisse";
    $msg = "Die Suche ergab [$total] Treffer";
    $junk = "";
    start_page($junk,$title,$msg);
    while ( list ($key,$hit) = each ($hits)) {
        if(!empty($hit)) {
            $LinkID = $hit["LinkID"];
            $LinkName = stripslashes($hit["LinkName"]);
            $LinkDesc = stripslashes($hit["Description"]);
            $LinkURL = stripslashes($hit["Url"]);
            $CatID = $hit["CatID"];
            $CatName = stripslashes($db->get_CatNames($CatID));
```

↓

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

```

        print "<DL>\n";
        print "<DT>
            <A HREF=\"\$LinkURL\" TARGET=\"_NEW\">
                \$LinkName</A>\n";
        print "<dd>\$LinkDesc</dd>\n";
        print "<dl><B>Found In:</B>&nbsp;
            <a href=\"\$PHP_SELF?
                viewCat=\$CatID\">\$CatName</a>\n";
        print "</dl>\n";
    }
} /* end while */
} /* end if */
echo "<p><hr>\n";
start_browse();
exit;
} else {
    // Restart
    start_page();
    start_browse();
    exit;
}
?>

```

Quelle auf der CD



Sie finden die vorgestellten Skripte fertig auf der CD im Pfad ANWENDUNGEN/PHPHOO. Dort finden Sie auch eine SQL-Datei mit den CREATE-Anweisungen.

Wie es weitergeht

Dieses Projekt wird in Kapitel 9 weiter ausgebaut, wo der Import eines Open Source-Verzeichnisses mit XML gezeigt wird – eine einfache Methode, um die Suchmaschine mit Links zu füllen. Mehr dazu finden Sie in ➡ Abschnitt 9.6.6 *Das XML-Projekt* ab Seite 711.

7.4 Zugriff auf ODBC-Quellen

Access ist weit verbreitet. Vor allem zum Datenimport und für Entwicklungsrechner ist der Zugriff mit PHP oft von Vorteil. Dieser Abschnitt zeigt, wie Sie mit Hilfe von ODBC auf Access-Datenbanken zugreifen können.

7.4.1 Grundlagen



Wer aus der Unix-Welt kommt und mit Windows-Programmen eher auf Kriegsfuß steht, mag den Sinn dieses Kapitels nicht sofort erkennen. Der praktische Grund liegt in der Natur der Software selbst. MS Access ist sicher kein Datenbankprogramm, das sich durch besondere Performance oder Sicherheit auszeichnet. Das spielt aber bei kleinen

Sites, die nur wenige Hits pro Tag haben, keine Rolle. Viel wichtiger erscheint der Umstand, dass Access auf nahezu jedem Server läuft. Statt aufwändiger Schnittstellen kann man einfach eine Datenbankdatei per FTP hochladen und schon ist die Datenbank verfügbar. Oft steht Access auch lokal zur Verfügung, da es Bestandteil der Professionell-Reihe der Microsoft Office-Pakete ist. So kann sich die Entwicklung entsprechender Skripte einfacher gestalten, als mit MySQL oder anderen Datenbanksystemen, die erst explizit installiert und konfiguriert werden müssten.

Der Einsatz von Access kann also durchaus Sinn machen. Die Anbindung ist von PHP aus über die ODBC-Schnittstelle möglich. Die Einrichtung auf einem Entwicklungssystem wird hier am Beispiel von Windows NT vorgestellt, wobei sich die Schritte ebenso auf Windows 2000 übertragen lassen. In Bezug auf die Verfügbarkeit von Access könnte auch Windows 9x/ME als Plattform dienen (vom Autor nicht getestet). Hier sollte der Leser sich aber nicht davon abhalten lassen, eigene Versuche zu unternehmen.

Die Nutzung von Access auf fremden Webservern ist ausgesprochen einfach. Die lokal entwickelte Anwendung wird per FTP auf den Server übertragen. Dort kann man dann per ODBC auf die Datenbank zugreifen. Einzige Voraussetzung ist ein entsprechender Treiber auf dem Server, der aber in aller Regel bei einem NT-Server zur Verfügung steht. Mit dem Transport der Datenbank werden auch die Daten übertragen. So werden umständliche Uploads und aufwändige Fernkonfigurationen vermieden.

Access bietet auch eine ausgesprochen einfache Oberfläche, die gestandene Windows-Nutzer leicht bedienen können. Da kann MySQL schon eine größere Hürde darstellen. Und diese Anleitung zeigt allgemein den Zugriff auf Datenbanken unter Windows per ODBC, sodass sich die Schritte auch gut für Foxpro, MS SQL Server 7/2000 oder andere Systeme nachvollziehen lassen.

Installationsvoraussetzungen

Für die folgenden Schritte benötigen Sie Zugriff auf einen Computer mit installiertem Access 97/2000, einem passenden ODBC-Treiber und natürlich installierten Access-Treibern (diese sind nach der Installation aber normalerweise schon fertig vorhanden). Sie sollten zumindest in der Lage sein, Access zu öffnen. Voraussetzung ist auch die Freigabe der ODBC-Funktionen in PHP, das heißt, entweder muss PHP mit ODBC-Funktionen kompiliert worden sein oder die entsprechende DLL ist vorhanden und kann dynamisch gebunden werden. Konsultieren Sie gegebenenfalls die Anleitung zu den Konfigurationseinstellungen in ➤ Kapitel 2 *Vorbereitung und Installation* ab Seite 45.

Übersicht über die ODBC-Funktionen

Die folgende Übersicht zeigt alle Befehle in diesem Abschnitt auf einen Blick:

- `odbc_autocommit`

Die Funktion wechselt den automatischen Bestätigungsmodus. normalerweise werden Änderungen nicht bestätigt.

- `odbc_binmode`

Hier werden binäre Daten behandelt.

- `odbc_close`

Die Funktion schließt eine ODBC-Verbindung.

- `odbc_close_all`

Hiermit werden alle ODBC-Verbindungen geschlossen.

- `odbc_commit`

Bestätigt alle ODBC-Transaktionen.

- `odbc_connect`

Verbindet mit einer Datenquelle und gibt ein Handle auf diese Verbindung zurück, das von allen anderen Funktionen verwendet wird.

- `odbc_cursor`

Ermittelt den Namen des Datenzeigers.

- `odbc_exec`

Die Funktion bereitet ein Kommando auf und führt es aus. `odbc_do` ist Synonym für `odbc_exec`.

- `odbc_execute`

Führt ein vorbereitetes Kommando aus.

- `odbc_fetch_into`

Holt eine Reihe der Ergebnisliste in ein Array.

- `odbc_fetch_row`

Die Funktion holt eine Reihe einer Ergebnisliste.

- `odbc_field_name`

Hiermit wird ein Spaltenname ermittelt.

- `odbc_field_type`
Ermittelt den Datentyp eines Feldes.
- `odbc_field_len`
Gibt die Länge (Breite) eines Feldes zurück.
- `odbc_free_result`
Gibt den von der Ergebnisliste verwendeten Speicher frei.
- `odbc_longreadlen`
Behandelt Spalten mit LONG-Daten (Große Binärfelder).
- `odbc_num_fields`
Gibt die Anzahl der Spalten in der Ergebnisliste zurück.
- `odbc_pconnect`
Öffnet eine persistente (ständige) Verbindung. Die Verbindung wird in folgenden Aufrufen anhand der Aufrufdaten wiedererkannt.
- `odbc_prepare`
Bereitet ein Kommando für die Ausführung vor. Das Kommando kann dann mehrfach schneller ausgeführt werden.
- `odbc_num_rows`
Ergibt die Anzahl der Zeilen in der Ergebnisliste.
- `odbc_result`
Holt die Daten aus der Ergebnisliste
- `odbc_result_all`
Gibt eine HTML-Tabelle mit Daten aus.
- `odbc_rollback`
Macht eine Transaktion rückgängig, wenn zuvor Operationen in einer Transaktion ausgeführt wurden.
- `odbc_setoption`
Setzt verschiedene ODBC-Optionen. Diese Funktion ist abhängig von der konkreten Datenquelle.

7.4.2 Beispielapplikation

Die Beispielapplikation zeigt ein Gästebuch, bei dem die Daten in einer MS Access-Datenbank gespeichert werden.

Vorbereiten der Datenbank

Im ersten Schritt legen Sie eine Access-Datenbank wie gewohnt an. Das folgende Beispiel zeigt ein einfaches Gästebuch mit einigen Administrationsfunktionen. Abbildung 7.6 zeigt die Struktur der Tabelle *guestbook*. Denselben Namen bekommt auch die Datenbank.

Abbildung 7.6:
Anlegen der Tabelle
guestbook

Feldname	Feldtyp	Beschreibung
id	AutoWert	Index
posted	Datum/Uhrzeit	Datum/Zeit der Veröffentlichung
name	Text	Name des Absenders (100)
email	Text	E-Mail (100)
location	Text	Ort (100)
message	Memo	Nachricht

ODBC-Schnittstelle aktivieren

Im nächsten Schritt wird die ODBC-Datenquelle angelegt. Dazu öffnen Sie die Systemsteuerung und das Programm DATENQUELLEN (ODBC). Fügen Sie nun eine neue System-DSN hinzu und wählen Sie als Treiber MICROSOFT ACCESS TREIBER aus.

Abbildung 7.7:
Vergeben eines
Datenquellennamens
und Verknüpfung
mit der Access-
Datenbank

Jetzt steht die Datenquelle auf dem System zur Verfügung und kann mit den ODBC-Funktionen von PHP angesprochen werden (Abbildung 7.8).

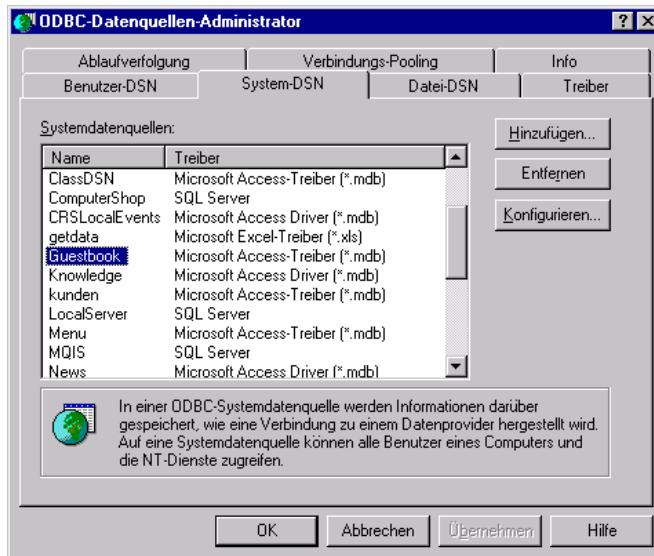


Abbildung 7.8:
Die DSN erscheint in
der Übersichtsliste

Es kann vorkommen, dass der Zugriff auf die Access-Datenbank verweigert wird. In diesen Fällen weisen Sie dem WebUser in Access die entsprechenden Rechte zu. Den in Abbildung 7.9 gezeigten Dialog erreichen Sie unter EXTRAS | ZUGRIFFSRECHTE.

Zugriffsprobleme



Abbildung 7.9:
Einstellungen der
Zugriffsrechte in
Access

Beispielapplikation

Die Applikation selbst realisiert ein einfaches Gästebuch. Auf die Erläuterung aller Details soll hier verzichtet werden, da hier nur die ODBC-Funktionen von Interesse sind.

Listing 7.14:
Gästebuchapplikation
mit Access-Zugriff

```
<?php
// Datenbankinformationen
$server="localhost"; // Serveradresse
$dsn = "Guestbook"; // ODBC DSN
$dbname="guestbook"; // Name der Tabelle
$uid=""; // Nutzernamen
$pwd=""; // Kennwort
$maxmsglength=1024; // maximale Nachrichtengröße
if($argv[0]=="admin"){ // Administration
?>
    <div align='center'>
    <form action="GuestbookAccess.php" method="POST">
    Loginname: <input type="Text" name="loginname">
    Kennwort: <input type="Password" name="pwd">
    <input type="Submit" value="Login">
    </form>
    </div>

<?php
} else { // normaler Betrieb
    if (!$action) { // Nur beim ersten Mal eintragen
        ?>
        <a href="GuestbookAccess.php#post">
            Tragen Sie sich ein!</a><br>
        <?php
    }
}

// Datenbank starten
$conn = odbc_connect($dsn,$uid,$pwd)
        or die( "Fehler beim Datenbankaufruf" );

// Wenn $autodelete, alte Einträge löschen
if($autodelete){
    $sql = "DELETE FROM $dbname WHERE posted < "
            . date("d-m-y H:i:s",(time()-($autodelete * 86400)));
    $result = odbc_exec($conn, $sql);
}

// Eintrag löschen
if($d!=""){
    $sql="DELETE FROM $dbname WHERE id=$d";
    if ($result = odbc_exec($conn, $sql)) {
        odbc_free_result($result);
    } else {
        echo "Löschen fehlgeschlagen.<br>\n";
    }
}

$isadmin = FALSE; // Administration?
```

```

if($loginname==$user && $pword==$password) $isadmin = TRUE;
// Neuer Eintrag ?
if($action=="Submit") {
    $name=$email=$company=$message=$loginname=$pword="";
    while(list($header,$value)=each($HTTP_POST_VARS)) {
        eval("$$.header.="\"$value\";");
    }
    if($maxmsglength && strlen($message)>$maxmsglength) {
        echo("<p>Die Nachricht ist leider zu lang!</p>\n");
    } elseif( strpos($email,"@")==FALSE ||
        strpos($email,".")==false) {
        echo "<p>Die E-Mail-Adresse ist ungültig".
            "Verwenden Sie die Form \"ihr_name@domain.de\".</p>\n";
    } else {
        if($name && $email && $message) {
            $test = odbc_exec($conn,
                "SELECT id FROM $dbname WHERE
                (name='$name' and email='$email')");
            if (odbc_fetch_row($test)==TRUE) {
                echo "Datensatz bereits vorhanden,
                    aktualisiere...<br>\n";
                $old_id = odbc_result($test,"id");
                $sql="UPDATE $dbname SET dateposted='".
                    date("d.m.y H:i:s",time())."',
                    name='$name', email='$email',
                    location='$location',
                    message='$message'
                    WHERE id=$old_id";
            } else {
                $sql="INSERT INTO $dbname
                    (dateposted, name, email, location, message) ".
                    "VALUES ('".date("d.m.y H:i:s", time())."',
                        '$name', '$email', '$location',
                        '$message')";
            }
            echo $sql;
            $result = odbc_exec($conn, $sql);
            if(!$result) {
                echo("Fehler!");
            } else {
                echo('<center><b>Vielen Dank für Ihren
                    Eintrag!</b></center>');
                // E-Mail-Nachricht
                if($notify){
                    $emailmessage="Ihr Gästebuch wurde verwendet:\n".
                        "Von: $name\nE-Mail: $email\n".
                        "Aus: $location\nNachricht:\n
                        $message\n\n";
                }
            }
        } else {

```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

```

?>
<p>Es wurden einige Felder nicht ausgefüllt!</p>
<?php
    }
    }
} // Ende "if action"
// Anzeige der Einträge
$sql="SELECT * FROM $dbname ORDER BY dateposted DESC";
if(($result = odbc_exec($conn, $sql))){
    $bzm = 1;
    while(odbc_fetch_row($result, $bzm)) {
        echo('<hr><p>');
        if($isadmin){
            echo "<b>ADMINISTRATION</b><br>\n";
        }
    }
?>
<form action="GuestbookAccess.php" method="POST">
<input type="hidden" name="loginname"
    value="<?php echo($loginname);?>">
<input type="hidden" name="password"
    value="<?php echo($password);?>">
<input type="hidden" name="d"
    value="<?php echo odbc_result($result,"id"); ?>">
<input type="Submit" value="Löschen"><br>
<?php }
    echo "<table border='0'>\n";
    echo "<tr><td align='right'>    <b>Name:</b> </td>";
    echo "<td>".odbc_result($result,"name")."    </td></tr>";
    echo "<tr><td align='right'><b>E-Mail:</b> </td>";
    echo "<td>".odbc_result($result,"email")."    </td> </tr>";
    echo "<tr><td align='right'><b>Aus:</b></td>";
    echo "<td>".odbc_result($result,"location")."</td></tr>";
    echo "<tr><td align='right'><b>Datum:</b></td>";
    echo "<td>";
    echo odbc_result($result,"dateposted"));
    echo "</td></tr>";
    echo "<tr><td align='right'><b>Message:</b></td>";
    echo "<td>".odbc_result($result,"message")."</td></tr>";
    echo "</table>";
    if($isadmin) echo('</form>');
    $bzm++;
}
odbc_free_result($result);
}else{
    echo("Fehler!");
}
// Datenbank schließen
odbc_close_all();
?>

<hr>
<a name="post"><b>Bitte tragen Sie sich in unser Gästebuch
ein!</b></a>

```

```
<form action="GuestbookAccess.php" method="POST">
<table border='0'>
  <tr valign='top'>
    <td align='right'>
      <br>Name:
    </td>
    <td>
      <br><input type="Text" name="name" size="40" maxlength="100">
    </td>
  <tr>
    <td rowspan='4'>
      Nachricht:<br>
      <textarea name="message" cols="40" rows="10" wrap="PHYSICAL">
      </textarea><br>
    </td>
    <td>
      <br><input type="Text" name="email" size="40" maxlength="40">
    </td>
  <tr>
    <td align='right'>
      <br>E-Mail:
    </td>
    <td>
      <br><input type="Text" name="location" size="40" maxlength="100">
    </td>
  <tr>
    <td align='right'>
      <br>Wohnort:
    </td>
    <td>
      <br>
      <input type="Text" name="location" size="40" maxlength="100">
    </td>
  <tr>
    <td align='right'>
      <br>
      <input type="Text" name="location" size="40" maxlength="100">
    </td>
    <td>
      <br>
      <input type="Text" name="location" size="40" maxlength="100">
    </td>
  <tr>
    <td align='center'>
      <input type="hidden" name="action" value="Submit">
      <input type="Submit" value="Eintragen">
      <input type="reset" value="Löschen ">
    </td>
  </tr>
</table>
</form>
<hr>
```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Erläuterungen

Wie es funktioniert Die wesentlichen Funktionen sind die Verbindung zur Datenbank und die Abfrage und Anzeige der Resultate. Für die Verbindung werden mehrere Variablen definiert, die Änderungen leichter handhabbar machen.

```
$server = "localhost"; // Serveradresse
$dsn    = "Guestbook"; // ODBC DSN
$dbname = "guestbook"; // Name der Tabelle
$uid    = "";          // Nutzernamen
$pwd    = "";          // Kennwort
```

Dann wird die Verbindung zur ODBC-Quelle aufgebaut:

```
$conn = odbc_connect($dsn, $uid, $pwd) or
die("Fehler beim Datenbankaufruf");
```

Die Anzeige der Datensätze erfolgt mit einer SELECT-Anweisung

```
$sql="SELECT * FROM $dbname ORDER BY dateposted DESC";
if(($result = odbc_exec($conn, $sql))){
    $bzm = 1;
```

In einer Schleife werden alle Datensätze ausgegeben (diese Darstellung ist von den Tabellenelementen und der Administrationsfunktion befreit):

```
while(odbc_fetch_row($result, $bzm)) {
    echo odbc_result($result,"name")."<br>";
    echo odbc_result($result,"email")."<br>";
    echo odbc_result($result,"location")."<br>";
    echo odbc_result($result,"message")."<br>";
    $bzm++;
}
```

Optional kann der Ergebnisspeicher am Ende freigegeben werden; wenn keine weiteren Operationen folgen, wird diese Funktion am Skriptende implizit aufgerufen

```
odbc_free_result($result);
```

Skripte und eine Musterdatenbank finden Sie auf der CD und auf der Website zum Buch.

Erläuterung der ODBC-Funktionen

odbc_connect()
odbc_pconnect()

Mit der Funktion `odbc_connect` wird eine Verbindung zu einer ODBC-Datenbank eröffnet. Die Funktion gibt entweder ein Handle auf diese Verbindung oder FALSE zurück.

```
$dbhandle = odbc_connect("DSN","User","Password");
```

Aufgrund dieser Eigenschaft kann auch hier die Auswertung des Fehlers und die Verknüpfung mit dem Befehl die erfolgen:

```
$dbhandle = odbc_connect("DSN","User","Password")  
or die("Fehler");
```

odbc_connect verfügt über einen vierten, optionalen Parameter, der das Verhalten des Datensatzzeigers (Cursors) steuert. Folgende Konstanten sind einsetzbar:

- SQL_CUR_USE_IF_NEEDED

Der Datensatzzeiger wird nur verwendet, wenn er von der Applikation angefordert wird.

- SQL_CUR_USE_ODBC

Der ODBC-eigene Zeiger wird verwendet; diese Option ist treiberabhängig.

- SQL_CUR_USE_DRIVER

Der Datensatzzeiger des Treibers wird genutzt. Dazu muss der Treiber diesen selbst erzeugen. Dies funktioniert nicht bei allen Datenbanken.

- SQL_CUR_DEFAULT

Der Standardwert – kann auch weggelassen werden.

Die Funktion odbc_pconnect arbeitet genau gleich, nur dass die Verbindung am Ende des Skripts nicht geschlossen wird. Wenn ein weiterer Skript mit odbc_pconnect eine Verbindung anfordert, wird zuerst überprüft, ob bereits eine Verbindung mit gleichartigen Verbindungsdaten besteht. Ist das der Fall, wird diese verwendet, andernfalls wird eine neue Verbindung aufgebaut. Diese Form ist immer zu bevorzugen, weil der Verbindungsaufbau schneller vonstatten geht.

Auf die Abarbeitung normaler Befehle hat eine persistente Verbindung keinen Einfluss. Wird PHP als CGI-Programm installiert, gibt es keinen Unterschied zu odbc_connect.



Hinweis

Die einfachste Methode, eine SQL-Anweisung abzusetzen, ist über die Funktion odbc_exec. Die Funktion gibt ein weiteres Handle zurück, mit dem Zugriff auf die Ergebnisliste besteht.

odbc_exec()

```
$result = odbc_exec($dbhandle,"SELECT * FROM guestbook");
```

Es ist sinnvoll, die SQL-Anweisung in eine Variable zu legen, um leichter das fertige Konstrukt prüfen zu können:

```
$sql = "SELECT * FROM guestbook";  
$result = odbc_exec($dbhandle, $sql);
```

Wenn keine Ergebnisse zu erhalten waren, wird FALSE zurückgegeben.

odbc_fetch_row() Mit `odbc_fetch_row` wird geprüft, ob der adressierte Datensatz vorhanden ist. Diese Funktion kann also gut zum Steuern einer Schleife eingesetzt werden:

```
while(odbc_fetch_row($result, $pointer)) {
```

Dabei ist *\$result* das Handle auf eine Ergebnisliste und *\$pointer* der Zeiger auf den ausgewählten Datensatz.

odbc_result() Die Übergabe der Daten aus der Ergebnisliste erfolgt mittels `odbc_result` (Abfrage einzelner Felder) oder `odbc_fetch_into` (Übertragung eines Datensatzes in ein Array):

odbc_fetch_into()

```
echo odbc_result($result, "name");

$ok = odbc_fetch_into($result, $row, &$resultset);
if ($ok) {
    echo $resultset[0];
    echo $resultset[1];
}
```

Weitere Informationen zu den einzelnen Funktionen finden Sie in der Referenz im Anhang.

7.5 MySQL über Access bedienen

Die Frage, warum man MySQL über Access bedienen sollte, ist sicher etwas ungewöhnlich. Die kryptische Oberfläche und die nicht eben komfortable Administration sind jedoch nicht gerade förderlich für eine produktive Entwicklung. Gerade die Eingabe von Testdaten kann aufwändig werden.



Access bietet dagegen eine ausgereifte und einfach zu bedienende Windows-Oberfläche, die zudem einen gewissen Bekanntheitsgrad hat.

7.5.1 Einrichtung



Der Trick besteht in der Nutzung des bereits in Kapitel 6 beschriebenen MySQL-ODBC-Treibers. Dabei wird für eine MySQL-Datenbank eine ODBC-Quelle angelegt. In Access kann man eine Datenbank mit einer ODBC-Quelle verbinden und alle Werkzeuge direkt auf diese Quelle anwenden. Dies kann dann auch eine MySQL-Datenbank sein.

Um dies zu tun, gehen Sie in folgender Reihenfolge vor:

- Legen Sie eine System-DSN für die MySQL-Datenbank an.
- Öffnen Sie Access und verbinden Sie sich mit der ODBC-Quelle.

- Nutzen Sie die Datenbank unter Access, als wäre es eine Access-Datenbank.

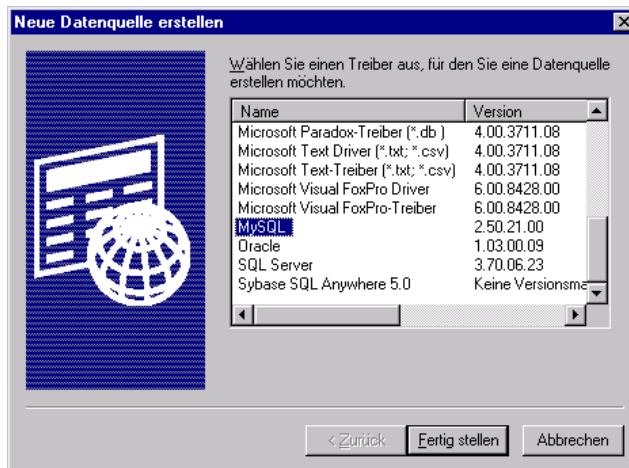


Abbildung 7.10:
Auswahl des
MyODBC-Treibers

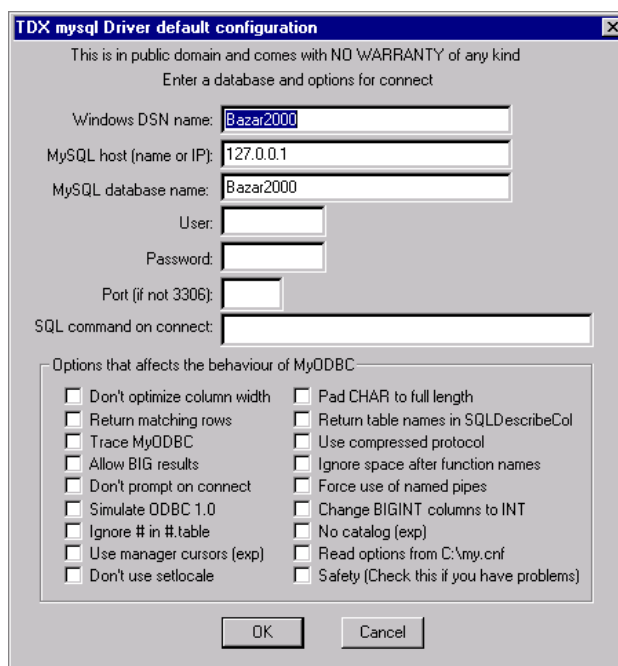
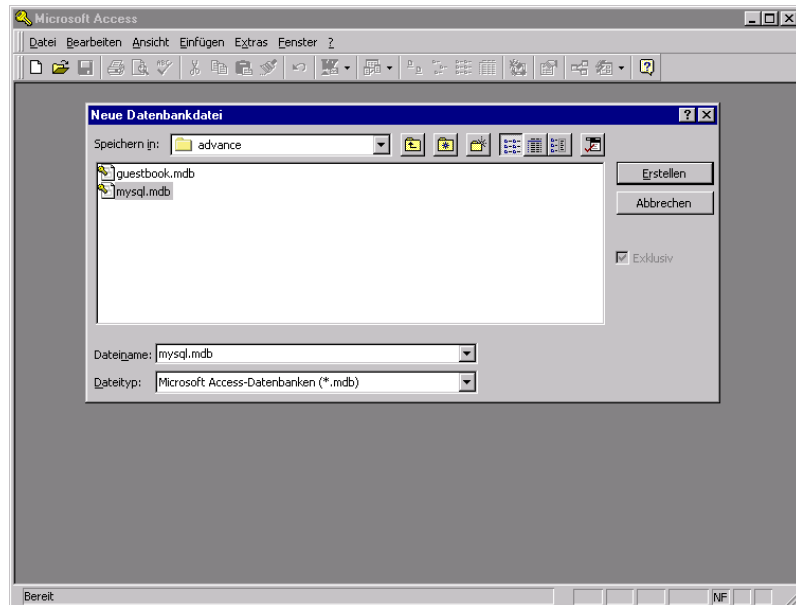


Abbildung 7.11:
Einstellungen der
Datenbank-
verbindungen

7.5.2 Nutzung

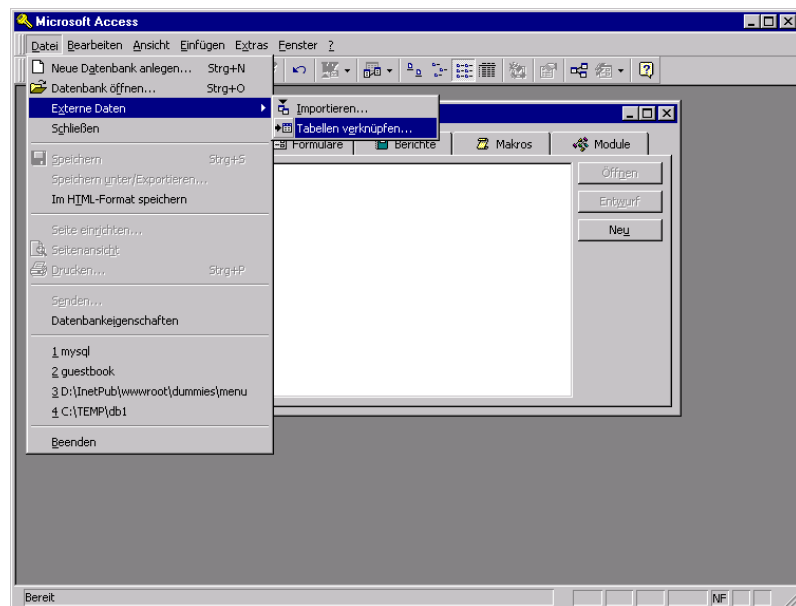
Nach der Verfügbarmachung von MySQL über ODBC können Sie Access direkt damit verbinden. Dazu legen Sie eine neue Datenbank mit DATEI | NEU an.

Abbildung 7.12:
Legen Sie zuerst eine
neue Datenbank an



Jetzt verknüpfen Sie die leere Datenbank mit der MySQL-Datenquelle.
Wählen Sie dazu DATEI | EXTERNE DATEN | TABELLEN VERKNÜPFEN.

Abbildung 7.13:
Verknüpfen mit der
externen Datenbank



In dem folgenden Dialog stellen Sie als Dateityp für die Anzeige ODBC ein. Das in Abbildung 7.14 gezeigte Dialogfeld öffnet sich automatisch und Sie können die zuvor angelegte MyODBC-Quelle auswählen.

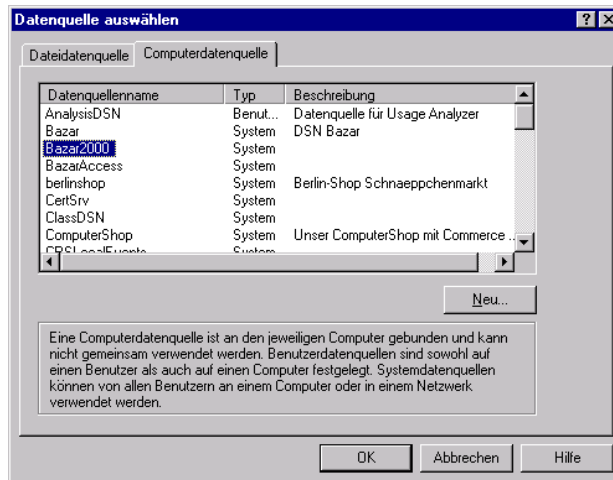


Abbildung 7.14:
Auswahl der
MySQL-Datenquelle

Im nächsten Schritt werden die Tabellen und die Spalten jeder einzelnen Tabelle ausgewählt. Sie können so Teile der Datenbank gezielt verfügbar machen. Abbildung 7.15 und Abbildung 7.16 zeigen die entsprechenden Dialoge.

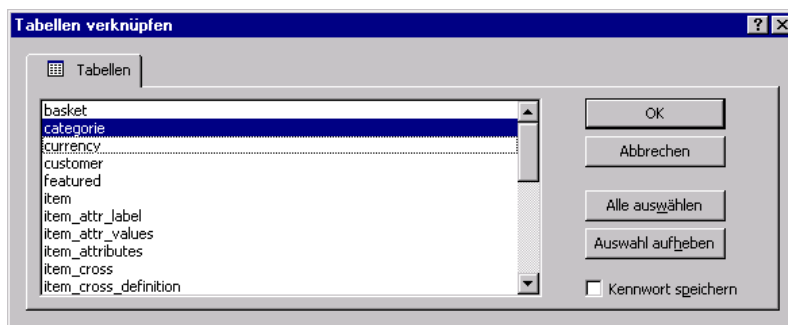


Abbildung 7.15:
Auswahl der Tabellen

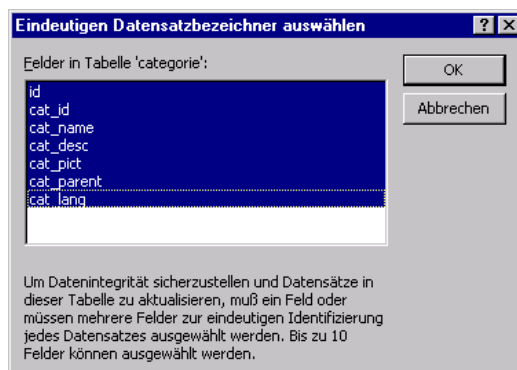
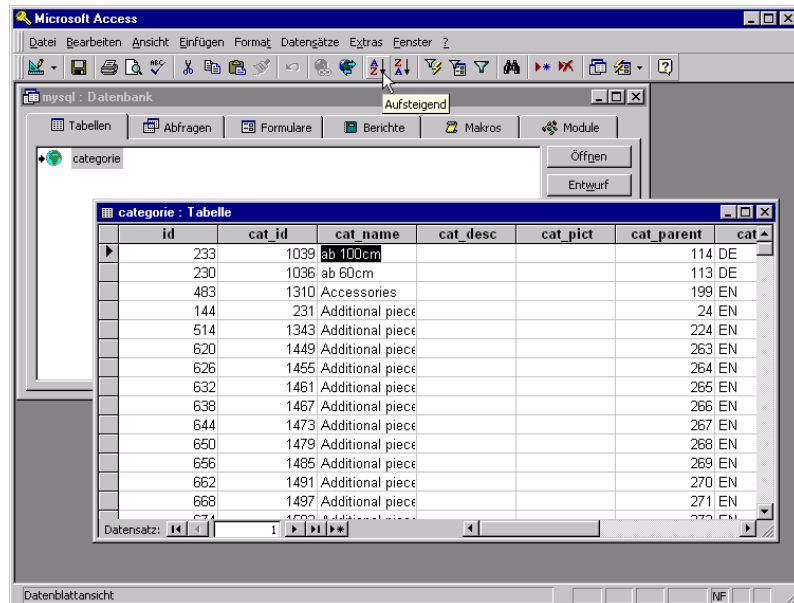


Abbildung 7.16:
Auswahl der Spalten
einer Tabelle

Sind alle Schritte absolviert, können Sie die verknüpfte Tabelle nun mit einem Doppelklick öffnen und direkt manipulieren, sortieren, Daten hinzufügen usw.

Abbildung 7.17:
Direkter Zugriff auf
MySQL über Access



7.6 Administration mit phpMyAdmin

phpMyAdmin ist das bekannteste Frontend für MySQL. Es ist sehr einfach zu installieren und zu benutzen. Auch im Webpace ist der Einsatz anstandslos möglich.

7.6.1 Einführung

**Beachten Sie die
©-Hinweise auf der
CD!**

Das Programm *phpMyAdmin* ist eine browserbasierte Oberfläche zur Bedienung von MySQL über das WWW, entwickelt von Tobias Ratschiller (*tobias@dnet.it*). Die Homepage erreichen Sie unter der folgenden Adresse:

<http://www.phpwizard.net>

Funktionen

Das Programm liegt derzeit in der Version 2.1.0 vor. Verwendet werden kann es mit MySQL ab Version 3.21.x, mit geringen Einschränkungen auch mit älteren Versionen. Folgende Funktionen sind per Mausklick ausführbar:

- Erzeugen und Löschen von Datenbanken
- Erzeugen, Löschen, Leeren und Kopieren von Tabellen
- Löschen, Bearbeiten und Hinzufügen von Feldern
- Ausführen von SQL-Kommandos und Stapelanweisungen
- Bearbeitung von Schlüsseln, Indizes usw.

- Export und Import von Text- und CVS²⁵-Dateien in die Datenbank
- Administration einer Datenbank

Wenn Sie Webservice mit MySQL-Unterstützung bei einem Provider anmieten, werden einige Funktionen (beispielsweise das Löschen von Datenbanken) möglicherweise aus Sicherheitsgründen gesperrt sein.

Installation

Die Installation ist ausgesprochen einfach und lohnt, auch wenn nur ein einziges Projekt mit MySQL umgesetzt werden soll. Gehen Sie einfach wie folgt vor:

Installation

- Entpacken der Distributionsdatei.
- Kopieren der Dateien in das entsprechende Verzeichnis des Webservers.
- Konfigurieren der Datei `CONFIG.INC.PHP`. Dies wird nachfolgend beschrieben.
- Starten des Skripts `INDEX.PHP`.

Falls Ihr Provider die Verknüpfung mit PHP über die Dateierweiterung `PHP3` herstellt, ist auch dafür eine Version mit entsprechenden Endungen verfügbar.



Die Konfigurationsdatei enthält einige Variablen, deren Werte vor dem ersten Start gesetzt werden müssen. Danach wird die Datei gespeichert und phpMyAdmin kann genutzt werden.

Die Konfigurationsdatei `CONFIG.INC.PHP3` enthält parallele Zugriffsmöglichkeiten für mehrere Hosts über die Indizes des Arrays `$cfgServers`. Normalerweise müssen Sie nur den ersten Teil mit dem Index `[0]` einstellen. Folgende Variablen sind wichtig:

Konfiguration

- `$cfgServers[n]['port']`
Portnummer des MySQL-Servers. Der Standardwert ist 3306 und muss nicht angegeben werden.
- `$cfgServers[n]['host']`
Der Name des Host-Computers. Hier kann fast immer `localhost` stehen.
- `$cfgServers[n]['adv_auth']`
Ein Boolescher Wert, der die Art der Authentifizierung bestimmt. Für die normale (Basic)-Authentifizierung tragen Sie `FALSE` ein,

²⁵ CVS steht für Comma Separated Value, kommaseparierte Textdateien.

sonst TRUE. Bei der erweiterten Authentifizierung können Sie sich über HTTP als MySQL-Benutzer anmelden. Dadurch sind auch Umgebungen administrierbar, in denen verschiedene Nutzer Zugang zu eigenen Tabellen haben, sich aber nicht gegenseitig stören dürfen.

- `$cfgServers[n]['user']`.

Benutzername für den MySQL-Server.

- `$cfgServers[n]['password']`

Kennwort für den MySQL-Server.

- `$cfgServers[n]['stduser']`

Nutzername für den Standardbenutzer bei Basic Authentication.

- `$cfgServers[n]['stdpass']`

Kennwort für den Standardbenutzer.

- `$cfgServers[n]['only_db']`

Wenn Sie hier den Namen einer Datenbank eintragen, wird nur diese eine Datenbank dem Nutzer angezeigt. Bei großen Servern mit vielen parallelen Datenbanken ist dies sehr bequem, da fremde Datenbanken auch dann aufgelistet werden, wenn eigentlich kein Zugriff besteht.

- `$cfgManualBase`

Der hier angegebene Pfad zeigt auf die Online-Version der MySQL-Datenbankdokumentation. Standardmäßig wird die Originalversion unter *www.mysql.com* genutzt. Sie sollten hier eine lokale Kopie nutzen.

- `$cfgPersistentConnections`

Stellen Sie diese Variable auf TRUE, wenn Sie persistente (ständige) Verbindungen nutzen möchten.

- `$cfgMysqladmin`

Pfad zu MYSQLADMIN. Damit werden die Daten vom Server erneut geladen.

- `$cfgConfirm`

Wenn diese Variable auf TRUE steht, wird bei Löschversuchen eine zusätzliche Sicherheitsabfrage erzeugt.

- `$cfgMaxRows`

Anzahl der Zeilen, die bei einer Abfrage angezeigt werden. Wenn die Ergebnisliste mehr Zeilen beinhaltet, werden Links zum Blättern angezeigt.

- `$cfgMaxInputsize`

Größe des Eingabefeldes für neue Datensätze.

- `$cfgBorder`

Rand der Tabelle in Pixel.

- `$cfgThBgColor`

Hintergrundfarbe der Überschriftenzeilen.

- `$cfgBgColorOne`

Erste Zeilenfarbe.

- `$cfgBgColorTwo`

Zweite Zeilenfarbe.

- `$cfgOrder`

Dieser Wert bestimmt die Sortierreihenfolge, entweder "DESC" für absteigende oder "ASC" für aufsteigende Sortierung.

- `$cfgShowBlob`

Nur wenn diese Variable auf TRUE steht, werden BLOB-Felder angezeigt.

- `$cfgShowSQL`

Wenn diese Variable auf TRUE steht, werden die von PHPMYADMIN erzeugten SQL-Abfragen im Kopf der Seite angezeigt.

- `$cfgColumnTypes`

Dieses Array enthält alle von MySQL angebotenen Datentypen. Diese Liste muss normalerweise nicht geändert werden.

- `$cfgFunctions`

Ein Array mit der von MySQL unterstützten Funktionsliste. Diese Liste muss normalerweise nicht geändert werden.

- `$cfgAttributeTypes`

Die Liste der Attribute für die Felddefinition. Diese Liste muss normalerweise nicht geändert werden.

In der Datei CONFIG.INC.PHP wird außerdem mit `require` die Sprachdatei eingebunden. Tauschen Sie hier den Eintrag `ENGLISH.INC.PHP` gegen

folgende Zeile aus, wenn Sie Ihr phpMyAdmin deutsch sprechen lassen möchten:

```
require("german.inc.php");
```

Problembehandlung

Einige Probleme treten erfahrungsgemäß immer wieder auf. Sie sollten vor den ersten Schritten mit *phpMyAdmin* die folgende Liste kurz überfliegen. Die möglichen Fehlermeldungen finden Sie in der Marginalspalte.

Access denied	Diese Meldung zeigt fehlende Zugriffsrechte an. Wenn Sie bei einem Provider im Webpace arbeiten, werden Sie nicht auf andere Datenbanken zugreifen können. Oft ist die eigene Datenbank fertig angelegt und kann nicht gelöscht werden. Möglicherweise haben Sie in der Konfigurationsdatei auch einen falschen Namen oder ein falsches Kennwort angegeben.
I can't insert rows SQL-Error	Wenn Sie Daten einfügen und diese Meldung erscheint, haben Sie vielleicht einen falschen Datentyp verwendet. Wenn Sie Tabellen anlegen und der Fehler erscheint (oder ein allgemeiner SQL-Fehler), fehlt bei den Datentypen VARCHAR, TEXT oder BLOB die Angabe der Feldgröße.
Wrong syntax near ... SQL-Error	Ein allgemeiner SQL-Fehler tritt auf, wenn Sie Feldnamen angeben, die in MySQL reservierte Wörter sind, beispielsweise <i>limit</i> . Das passiert normalerweise nur beim Anlegen oder Bearbeiten von Tabellen.
Inhalt wird nicht bearbeitet	Wenn sich <i>phpMyAdmin</i> weigert, den Inhalt einer Tabelle zu bearbeiten, haben Sie vergessen, einen Primärschlüssel anzulegen.

7.6.2 Funktionsübersicht

Die folgende Übersicht zeigt die wichtigsten Funktionen und verschafft einen Eindruck vom Programm, wenn Sie *phpmyAdmin* noch nicht kennen. Dem Programm liegt aber auch ein ausführliches Handbuch bei.

Datenbanken und Tabellen anlegen

Legen Sie zuerst eine neue Datenbank an, wenn Sie ein Projekt mit *phpMySQL* starten. Wenn Sie auf ein vorhandenes Projekt aufsetzen oder einen Server mit Datenbank gemietet haben, können Sie dagegen gleich mit der Erzeugung und Bearbeitung der Tabellen beginnen. Abbildung 7.18 zeigt den Startdialog mit der Möglichkeit, eine neue Datenbank anzulegen. Links sehen Sie die Liste aller Datenbanken.

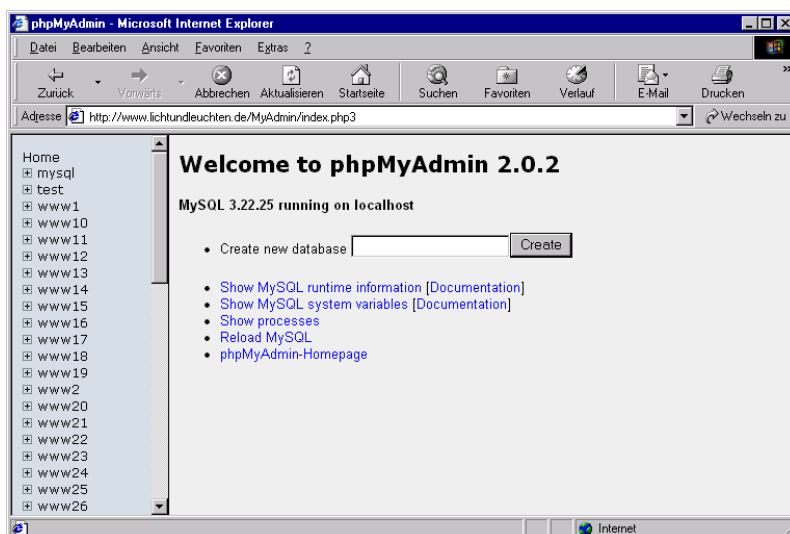


Abbildung 7.18:
Startbild von
phpMyAdmin

Abbildung 7.19 zeigt die Generierung einer neuen Datenbank. Die Datenbank erscheint in der linken Liste und kann nun mit einem Klick darauf bearbeitet werden. Wenn Sie auf die +-Zeichen klicken, werden die vorhandenen Tabellen angezeigt.

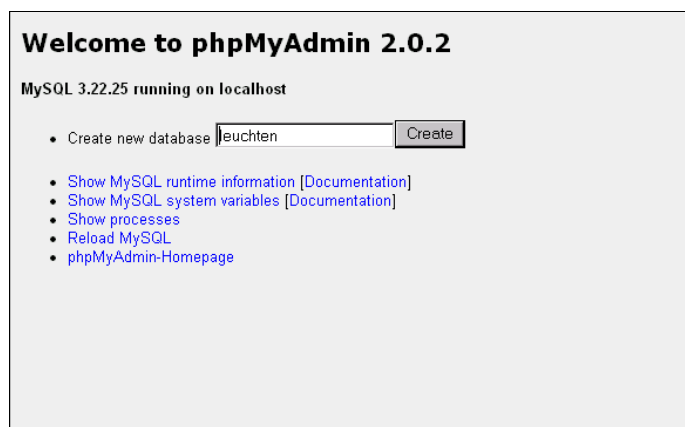


Abbildung 7.19:
Anlegen einer neuen
Datenbank

Abbildung 7.20 zeigt den leeren Kontrolldialog einer Datenbank. Sie können hier:

- *SQL-Anweisungen absenden.* Damit sind Anweisungen möglich, für die kein vorgefertigter Befehl zur Verfügung steht.
- *Datenstrukturen anzeigen.* Damit haben Sie Zugriff auf die Tabellenstruktur in Form einer CREATE TABLE-Anweisung.
- *Neue Tabellen anlegen.* Geben Sie dazu Name und Anzahl der Felder der Tabelle ein. Sie können später weitere Felder hinzufügen oder welche löschen.

Abbildung 7.20:
Eine leere Datenbank

Database www50

No tables found in database.

- Run SQL query/queries on database www50 [\[Documentation\]](#):
- [Query by Example](#)
- View dump (schema) of database
 - ☒ Structure only ☐ Add 'drop table'
 - ☐ Structure and data ☐ send
- Create new table on database www50:
 - Name:
 - Fields:
- [Drop database www50](#)

Abbildung 7.21 zeigt den Dialog zur Eingabe der Tabelleninformationen. Sie können jeden Eintrag später modifizieren. *phpMyAdmin* erzeugt an dieser Stelle eine SQL-Anweisung und sendet sie an die Datenbank.

Abbildung 7.21:
Dialog für eine neue
Tabelle mit einer
Anzahl Spalten

Database www50 - table tbl_product

Field	Type	Length/Set	Attributes	Null	Default	Extra	Primary	Index	Unique
id	INT			not null		auto_increment	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
name	VARCHAR	80		not null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
description	VARCHAR	255		null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
sku	VARCHAR	16		null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
category	INT			not null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
type	INT			null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
currency1	INT			null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
currency2	INT			null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
price	FLOAT			null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
vat	FLOAT			null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
layout	INT			null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
img_thumbnail	VARCHAR	255		null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
img_big1	VARCHAR	255		null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
img_big2	VARCHAR	255		null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
img_big3	VARCHAR	255		null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
img_big4	VARCHAR	255		null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
img_big5	VARCHAR	255		null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
img_big6	VARCHAR	255		null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
img_big7	VARCHAR	255		null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
img_big8	VARCHAR	255		null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
specialprice	TINYINT		BINARY	null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
linktip	INT			not null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
lang	CHAR	2		not null			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

[\[Documentation\]](#)

Wurde die Tabelle fertig eingegeben, klicken Sie auf **SAVE**, um die **CREATE TABLE**-Anweisung zu erzeugen und an die Datenbank zu senden. Abbildung 7.22 zeigt den Vorgang, wenn er erfolgreich beendet wurde. Treten an dieser Stelle Fehler auf, beachten Sie die Fehlerhinweise in ➞ Abschnitt *Problembehandlung* auf Seite 590. Sie können beliebig oft zum Eingabedialog zurückkehren und den Vorgang wiederholen. Solange Fehler auftreten, wird MySQL keine Änderungen an der Datenbank vornehmen.

Database www50 - table tbl_product

table tbl_product has been created.

SQL-query:
 CREATE TABLE tbl_product (id INT not null AUTO_INCREMENT, name VARCHAR (80) not null , description VARCHAR (255) , sku VARCHAR (16) , category INT not null , type INT , currency1 INT , currency2 INT , price FLOAT , vat FLOAT , layout INT , img_thumbnail VARCHAR (255) , img_big1 VARCHAR (255) , img_big2 VARCHAR (255) , img_big3 VARCHAR (255) , img_big4 VARCHAR (255) , img_big5 VARCHAR (255) , img_big6 VARCHAR (255) , img_big7 VARCHAR (255) , img_big8 VARCHAR (255) , specialprice TINYINT UNSIGNED , linktip INT not null , lang CHAR (2) not null , PRIMARY KEY (id))

Field	Type	Attributes	Null	Default	Extra	Action
id	int(11)		No	0	auto_increment	Change Drop Primary Index Unique
name	varchar(80)		No			Change Drop Primary Index Unique
description	varchar(255)		Yes			Change Drop Primary Index Unique
sku	varchar(16)		Yes			Change Drop Primary Index Unique
category	int(11)		No	0		Change Drop Primary Index Unique
type	int(11)		Yes			Change Drop Primary Index Unique
currency1	int(11)		Yes			Change Drop Primary Index Unique
currency2	int(11)		Yes			Change Drop Primary Index Unique
price	float(10,2)		Yes			Change Drop Primary Index Unique
vat	float(10,2)		Yes			Change Drop Primary Index Unique
layout	int(11)		Yes			Change Drop Primary Index Unique
img_thumbnail	varchar(255)		Yes			Change Drop Primary Index Unique
img_big1	varchar(255)		Yes			Change Drop Primary Index Unique
img_big2	varchar(255)		Yes			Change Drop Primary Index Unique
img_big3	varchar(255)		Yes			Change Drop Primary Index Unique
img_big4	varchar(255)		Yes			Change Drop Primary Index Unique
img_big5	varchar(255)		Yes			Change Drop Primary Index Unique
img_big6	varchar(255)		Yes			Change Drop Primary Index Unique
img_big7	varchar(255)		Yes			Change Drop Primary Index Unique
img_big8	varchar(255)		Yes			Change Drop Primary Index Unique
specialprice	tinyint(3)	UNSIGNED	Yes			Change Drop Primary Index Unique
linktip	int(11)		No	0		Change Drop Primary Index Unique
lang	char(2)		No			Change Drop Primary Index Unique

Keyname Unique Field Action
 PRIMARY Yes id [Drop](#)
[\[Documentation\]](#)

- [Browse](#)
- [Select](#)
- [Insert](#)
- Add new field: [Go](#)
- [Insert textfiles into table](#)
- View dump (schema) of table
 - ☒ Structure only ☐ Add 'drop table' [Go](#)
 - ☐ Structure and data ☐ send
 - ☐ CSV data terminated by:
- Rename table to: [Go](#)
- Copy table to:
 - ☒ Structure only ☐ Structure and data [Go](#)

Abbildung 7.22:
Eine Tabelle wurde
erfolgreich erzeugt

Legen Sie nun alle folgenden Tabellen auf dieselbe Art und Weise an. Wenn Sie in der linken Leiste auf den Namen der Datenbank klicken, sehen Sie das Übersichtsfenster mit den Tabellen und verschiedenen Optionen (Abbildung 7.23).

Abbildung 7.23:
Übersicht einer
Datenbank mit
Tabellen

Database www50

table	Action	Records
tbl_basket	Browse Select Insert Properties Drop Empty	0
tbl_cat	Browse Select Insert Properties Drop Empty	0
tbl_customer	Browse Select Insert Properties Drop Empty	0
tbl_product	Browse Select Insert Properties Drop Empty	0
tbl_topten	Browse Select Insert Properties Drop Empty	0

• Run SQL query/queries on database www50 [Documentation]:

• Query by Example

• View dump (schema) of database

☒ Structure only
 ☐ Add 'drop table'

☐ Structure and data
 ☐ send

• Create new table on database www50:

Name:

Fields:

• Drop database www50

Sie können die Tabellen nun einzeln bearbeiten, indem die entsprechenden Optionen angeklickt werden:

- BROWSE
Zeigt alle Daten an (SELECT * FROM *table*).
- SELECT
Führt ein SELECT *feldlist* FROM *table* aus, wobei Sie die Feldliste und eine zusätzliche WHERE-Bedingung definieren können.
- INSERT
Fügt Daten ein.
- PROPERTIES
Zeigt alle Eigenschaften der Tabelle an.
- DROP
Löscht die Tabelle.
- EMPTY
Leert die Tabelle (Struktur bleibt erhalten).

Tabellen bearbeiten

Haben Sie eine Tabelle bereits erzeugt, können Sie leicht die Struktur bearbeiten. Abbildung 7.24 zeigt den Dialog; folgende Optionen sind möglich:

- CHANGE

Ändern der Struktur.

- DROP

Löschen der Tabelle.

- PRIMARY

Primärschlüssel festlegen.

- INDEX

Index festlegen.

- UNIQUE

Festlegen von Spalten, deren Zellen einen eindeutigen Inhalt haben müssen.

Database www50 - table tbl_basket

Field	Type	Attributes	Null	Default	Extra	Action
id	int(11)		No	0		Change Drop Primary Index Unique
customer	int(11)		No	0		Change Drop Primary Index Unique
product	int(11)		No	0		Change Drop Primary Index Unique
qty	int(11)		No	0		Change Drop Primary Index Unique
price	double(16,4)		No	0.0000		Change Drop Primary Index Unique
orderdate	datetime		No	0000-00-00 00:00:00		Change Drop Primary Index Unique
sessionid	varchar(200)		No			Change Drop Primary Index Unique
ordered	tinyint(4)		No	0		Change Drop Primary Index Unique
delivered	tinyint(4)		No	0		Change Drop Primary Index Unique
garbage	timestamp(14)		Yes			Change Drop Primary Index Unique

Keyname Unique Field Action

PRIMARY Yes id Drop

id No id Drop

[Documentation]

- [Browse](#)
- [Select](#)
- [Insert](#)
- Add new field:
- [Insert textfiles into table](#)
- View dump (schema) of table
 - ☒ Structure only ☐ Add 'drop table'
 - ☐ Structure and data ☐ send
 - ☐ CSV data terminated by:
- Rename table to:
- Copy table to:
 - ☒ Structure only ☐ Structure and data

Abbildung 7.24:
Dialog zum
Bearbeiten einer
Tabelle

Der Unterschied zwischen Browse und Select besteht in der Möglichkeit, bei Select selbst die Felder und den Inhalt der WHERE-Bedingung zu bestimmen (siehe Abbildung 7.25). Sie können dabei:

- die Felder bestimmen,
- die WHERE-Bedingung direkt angeben oder
- jedes Feld anhand einer Einschränkung auswählen.

Klicken Sie auf GO, um die Abfrage zu starten.

Abbildung 7.25:
Generierung einer
SELECT-Anweisung

Database www50 - table tbl_basket

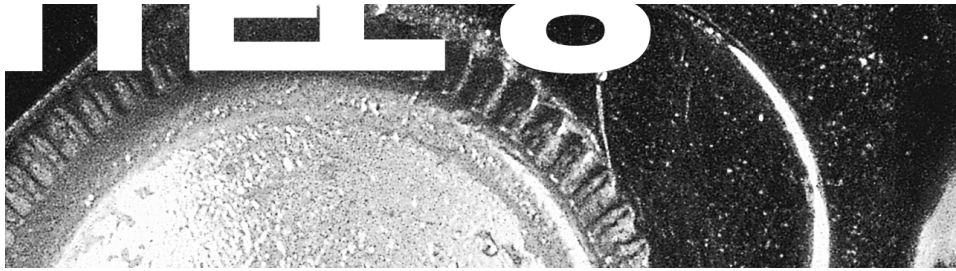
Select fields (at least one):

- id
- customer
- product
- qty
- price
- orderdate
- sessionid
- ordered
- delivered
- garbage

• Add search conditions (body of the "where" clause):
 [\[Documentation\]](#)

• Do a "query by example" (wildcard: "%")

Field	Type	Value
id	int	<input type="text"/>
customer	int	<input type="text" value="10000"/>
product	int	<input type="text"/>
qty	int	<input type="text"/>
price	real	<input type="text"/>
orderdate	datetime	<input type="text"/>
sessionid	string	<input type="text"/>
ordered	unknown	<input type="text"/>
delivered	unknown	<input type="text"/>
garbage	timestamp	<input type="text"/>



Praxis II – Datenbanklösungen

Dieses Kapitel bietet viele kleine und größere Projekte, die sich im Alltag sofort einsetzen lassen. Sie können alle Skripte anpassen und frei verwenden. Wichtige Bestandteile werden erläutert.

Kleine Dienstprogramme (Seite 599)

Gästebuch (Seite 609)

Authentifizierung mit Datenbank (Seite 612)

Umfrage (Seite 614)

8.1 Kleine Dienstprogramme

Dieser Abschnitt erfüllt zwei Aufgaben. Zum einen wird anhand einer Reihe von Skripten gezeigt, wie PHP-Programme aussehen. Zum anderen wird für das eine oder andere Projekt vielleicht eine fertige Funktion zu finden sein, die eigenen Aufwand erspart.

8.1.1 Übersicht

Die folgenden Programme wurden aus verschiedenen Quellen im Internet zusammengetragen und teilweise ergänzt oder neu programmiert. Beachten Sie die Copyright-Hinweise in den Skripten. Die meisten Anwendungen in diesem Teil stützen sich auf eine Datenbank. Verwendet wird hier grundsätzlich MySQL.

- Tabellen anzeigen
- Datumsbehandlung mit MySQL
- Eine MySQL-Tabelle abfragen und ausgeben
- Kreditkartendaten überprüfen (Prüfsummencheck)
- Eine WHERE-Bedingung aus einer Suchanfrage generieren.

Vorbereitung

Die Skripte, die im folgenden auf eine Datenbank zugreifen, schließen in jedem Fall die Datei `OPEN.INC.PHP4` ein. Diese Datei liegt im Verzeichnis `ANWENDUNGEN` und sollte angepasst werden, wenn Sie nicht die Standardeinstellungen verwenden oder nicht lokal arbeiten. Außerdem wird die bereits in Kapitel 7 verwendete Datenbank *customers* und die Tabelle *address* verwendet.

8.1.2 Tabellen anzeigen

Die folgende kleine Applikation zeigt eine Tabelle aus einer MySQL-Datenbank an und erlaubt durch anklicken der Titelleiste das Sortieren der Spalte. Es wird das in Kapitel 7 gezeigte Skript `OPEN.INC.PHP4` verwendet, um den Zugriff auf die Datenbank zu realisieren.

```
<?php
include("scripts/open.inc.php4");
$db = "customers";
@mysql_select_db ($db) or die ("Kann Datenbank nicht finden");
if(!isset($sort)) {
    $strQuery = "SELECT * FROM address";
} else {
    $strQuery = "SELECT * FROM address ORDER BY $sort $dir";
```

Listing 8.1:
Universelle
Abruffunktion für
Tabellen: [*showtable*](#)


```

}
$result = mysql_query($strQuery, $conn);
echo '<table border="1"><tr>';
while($field = mysql_fetch_field($result)) {
    echo <<<TABLE
    <th nowrap class="small">
    <a href="$PHP_SELF?sort=$field->name&dir=ASC">+</a>
    $field->name
    <a href="$PHP_SELF?sort=$field->name&dir=DESC">-</a>
    </th>
TABLE;
}
echo '</tr>';
while($row = mysql_fetch_array($result)) {
    echo '<tr>';
    for($i=0; $i < mysql_num_fields($result); $i++) {
        echo "<td class=small>$row[$i]</td>";
    }
    echo '</tr>';
}
echo '</table>';
?>

```

Wie es funktioniert Die Einleitung mit der Öffnung der Datenbankverbindung dürfte Ihnen inzwischen geläufig sein. Der Tabellennamen wird in der Variablen *\$table* übergeben.

Die Abfrage wird entweder unsortiert oder nach einer Spalte sortiert ausgeführt, je nach Aufruf des Skripts:

```

if(!isset($sort)) {
    $strQuery = "SELECT * FROM address";
} else {
    $strQuery = "SELECT * FROM address ORDER BY $sort $dir";
}
$result = mysql_query($strQuery, $conn);

```

Interessant ist die Abfrage der Feldnamen zum automatischen Erstellen der Kopfzeile:

```
while($field = mysql_fetch_field($result)) {
```

Die Links zum Sortieren werden hier ebenfalls generiert:

```

echo <<<TABLE
<th nowrap class="small">
<a href="$PHP_SELF?sort=$field->name&dir=ASC">+</a>
$field->name
<a href="$PHP_SELF?sort=$field->name&dir=DESC">-</a>
</th>
TABLE;

```

Dann folgt die Schleife zum Abrufen der Datensätze

```
while($row = mysql_fetch_array($result)) {
```

und innerhalb der Schleife werden alle Felder abgerufen und angezeigt:

```
for($i=0; $i < mysql_num_fields($result); $i++) {
    echo "<td class=small>$row[$i]</td>";
}
```

custID	fname	sname	company	street	zip	city	telephone	telefax	email
13	Nebel	Herbert		Mommensenstrasse	10629	Berlin	030/6814244		nebel@comvalley.com
12	Neumeier	Frank	Ziff Davis Verlag	Riesstrasse	80992	München	089/14312142	089/14312100	neumeier@ziff.de
11	Meene	Klaus	Living-Systems	Roggenbachstrasse	78050	Villingen	07721/98910	07721/989191	meene@living-systems.de
10	Meissner	Bernd	Openshop Holding AG	Wilhelmstrasse	89073	Ulm	0731/1553-100	0731/1552-111	info@openshop.de
9	Eisenmenger	Frank	CIB	Robert-Rössle-Straße	13125	Berlin	030/94892911	030/94892912	cib@mdc-berlin.de
8	Kallola	Dan	e-com AG	Richard-Wagner-Strasse	70184	Stuttgart	0711/24894812	0711/24894811	info@ecomag.de
7	Gaub	Sissi	e-com AG	Richard-Wagner-Strasse	70184	Stuttgart	0711/24894812	0711/24894811	info@ecomag.de
6	Fernau	Peter	e-com AG	Richard-Wagner-Strasse	70184	Stuttgart	0711/24894812	0711/24894811	fernau@ecomag.de
5	Jakob	Georg	Compuserve	Hauptstrasse	82008	Unterhaching	089/66571160	089/66571316	jakob@compuserve.com
4	Borgmann	Georg	Euro Event	Tronjestrasse	44319	Dortmund	0231/212110	0231/21929	nomail
3	Schellon	Peter	Euro Event	Tronjestrasse	44319	Dortmund	0231/212110	0231/21929	nomail
2	Dannegger	Christian	Living-Systems	Roggenbachstrasse	78050	Villingen	07721/98910	07721/989191	danneger@living-systems.de
1	Hopstein	Alexander	Conga Communications	Börselgasse	22765	Hamburg	040/1234356	040/8976541	hopstein@conga.de

Abbildung 8.1:
Ausgabe der
Mustertabelle mit
Sortierung

8.1.3 Datumsbehandlung

Die hier vorgestellte kleine Funktion fragt ein HTML-Formularfeld ab und wandelt ein darin enthaltenes Datum so um, dass es problemlos in das Datumsfeld einer MySQL-Tabelle eingefügt werden kann. Das interne Datumsformat in MySQL ist (ohne Zeitangaben):

```
YYYY-MM-DD
```

Folgende Eingabeformate sind zulässig:

- d.m.y (d = Tag, m = Monat, y = Jahr)
- y.m.d
- d.m
- d

Fehlt einer der Bestandteile des Datums, wird diese Information dem aktuellen Systemdatum entnommen. Wenn die Funktion eine leere Zeichenkette zurückgibt, konnte das Eingabedatum nicht ausgewertet werden.

Die Funktion kann folgendermaßen benutzt werden:

```
$date_ok = input2date($date_from_form_field);
```

```
<?php
function input2date($idate) {
    $token="-./ ";
    $p1 = strtok($idate, $token);
    $p2 = strtok($token);
    $p3 = strtok($token);
    $p4 = strtok($token);
```

Listing 8.2:
MySQL-gerechte
Umwandlung von
Datumswerten:
mysqldata

```

$date = $y = $m = $d = '';
// prüfe 'd.m.y'
if (($p1>0 && $p1<32) && ($p2>0 && $p2<13) && ($p3>32)) {
    $y = $p3;
    $m = $p2;
    $d = $p1;
} elseif ($p1>0 && ($p2>0 && $p2<13) && ($p3>0 && $p3<32)) {
    // prüfe y.m.d (wenn d.m.y ansonsten sinnlos ist)
    $y = $p1;
    $m = $p2;
    $d = $p3;
}
// prüfe 'd.m'
if ($y == "" && ($p3=="") && ($p2>0 && $p2<13) &&
    ($p1>0 && $p1<32)) {
    $y = date("Y");
    $m = $p2;
    $d = $p1;
}
// prüfe 'd'
if ($y == "" && ($p3=="") && ($p2=="") && ($p1>0 && $p1<32)) {
    $y = date("Y");
    $m = date("m");
    $d = $p1;
}
// hinzufügen von 1900 oder 2000 zur Jahreszahl
if ($y!="" && $y<=99) {
    if ($y>=70) $y = $y + 1900;
    if ($y<70) $y = $y + 2000;
}
if ($y!="") {
    if (checkdate($m, $d, $y)) $date="$y-$m-$d";
}
return $date;
}
?>

```

Wie es funktioniert Das Beispiel nutzt intensiv die Funktion `strtok`. Beachten Sie, dass die Funktion bei jedem erneuten Aufruf das *nächste* Element in der Zeichenkette sucht. Der folgende Teil zerlegt die Zeichenkette tatsächlich in vier Teile:

```

$token="-./ ";
$p1 = strtok($idate, $token);
$p2 = strtok($token);
$p3 = strtok($token);
$p4 = strtok($token);

```

Der Rest des Skripts prüft jeweils alle erlaubten Kombinationen mit einzelnen `if`-Anweisungen. Standardmäßig wird auf Datumsangaben der Form »d.m.y« geprüft, alternativ auf »y.m.d«. Letzteres aber nur, wenn die Standardform keine sinnvolle Angabe ergibt. Sie müssen,

wenn zweistellige Jahreszahlen akzeptiert werden, hier eine Priorität festlegen, weil ansonsten Daten wie 13.3.03 nicht korrekt erkannt werden können (dieses Datum kann bei der im Deutschen üblichen Leseweise der 13. März 2003 sein, im englischen Sprachraum dagegen der 3. März 2013).

Die Behandlung zweistelliger Jahreszahlen ist leicht zu modifizieren:

```
if ($y!="" && $y<=99) {  
    if ($y>=70) $y = $y + 1900;  
    if ($y<70) $y = $y + 2000;  
}
```

Sie können hier den Wert 70 anpassen. In der vorgestellten Version werden Daten größer oder gleich 70 zu 19xx gewandelt, kleinere Daten zu 20xx.

8.1.4 Eine SELECT-Liste aus MySQL erstellen

Das folgende Skript liest Werte aus einer MySQL-Tabelle und generiert mit diesen Werten ein <SELECT>-Tag. Die Aufgabe wird von der Funktion *buildselect* erledigt, der übrige Teil des Skripts dient der Demonstration.

```
<?php  
function buildselect($result, $field, $name,  
                    $multiple = '', $size = 1)  
{  
    echo "<select $multiple name=\"$name\" size=$size>";  
    $i=0;  
    while($row = mysql_fetch_assoc($result)) {  
        echo "<option value=\"$row[$field]\" . \">";  
        echo $row[$field];  
        echo "</option>\n";  
    }  
    mysql_data_seek($result, 0);  
    echo "</select>";  
}  
include("scripts/open.inc.php4");  
$db = "customers";  
@mysql_select_db ($db) or die ("Kann Datenbank nicht finden");  
$result = mysql_query("SELECT * FROM address", $conn);  
buildselect($result, 'sname', 'vorname');  
buildselect($result, 'fname', 'nachname');  
buildselect($result, 'company', 'firma', '', 4);  
?>
```

Listing 8.3:
<SELECT>-Liste aus
einer MySQL-Tabelle
erstellen: buildselect

Dieses sehr einfache Beispiel löst ein häufig lästiges Problem. Die Funktion erzeugt die Liste auf der Basis eines Ergebnis-Handles einer Datenbankabfrage. Die Eigenschaften des Tags werden in mehreren

Wie es funktioniert

Variablen übergeben, wobei die Parameter *\$multiple* und *\$size* optional sind.

```
echo "<select $multiple name=\"\$name\" size=$size>";
```

Innerhalb einer while-Schleife werden die Inhalte abgefragt:

```
while($row = mysql_fetch_assoc($result)) {
```

Die Feldinhalte bilden sowohl den Inhalt (value) als auch den Anzeigewert.

```
echo "<option value=\"\" . $row[$field] . \">";
echo $row[$field];
echo "</option>\n";
```

Damit das Ergebnislisten-Handle weiter verwendet werden kann, wird es mit der folgenden Anweisung in den ursprünglichen Zustand zurückversetzt:

```
mysql_data_seek($result, 0);
```

Die Benutzung erfolgt durch einen Aufruf der Funktion, analog den buifolgenden Beispielen:

```
buildselect($result, 'sname', 'vorname');
buildselect($result, 'fname', 'nachname');
buildselect($result, 'company', 'firma', '', 4);
```

Abbildung 8.2:
Ausgabe mit Daten
aus der Mustertabelle

Das eigentliche Formular mit Absendeschaltfläche ist im Beispiel nicht enthalten – freilich sind derartige Listen nur in Formularen sinnvoll einsetzbar.

8.1.5 Kreditkartencheck

Die Funktion prüft Kreditkartennummern auf Vollständigkeit und korrekte Prüfziffer. Dies erlaubt zumindest schon einen ersten Vorabcheck auf Tippfehler oder offensichtliche Betrugsversuche bei einem Shop. Sie können damit vermeiden, dass allzu viele fehlerhafte Abfragen an ein Zahlungsgateway gesendet werden, was unter Umständen auch Kosten sparen kann.

Die Funktion verwendet den Mod 10-Algorithmus zur Prüfung. Die Funktion wird folgendermaßen verwendet:

```
$ok = validateCC($number, $type);
```

Die Variable *\$number* enthält die zu prüfende Kreditkartennummer, *\$type* den Typ der Kreditkarte (»visa«, »mastercard«, »discover« oder



Beispiel

»amex«). Die Funktion gibt im Erfolgsfall TRUE zurück, sonst FALSE oder -1, wenn der Kartentyp nicht korrekt ist.

```
<?php
function validateCC($ccnum, $type = ''){
    $type = strtolower($type);
    $ccnum = preg_replace( '/[-[:space:]]/', ' ', $ccnum);
    if (strlen($ccnum) == 0) return FALSE;
    switch ($type) {
        case '':
            break;
        case 'visa':
            if (strlen($ccnum) != 13 and strlen($ccnum) != 16
                or substr($ccnum, 0, 1) != "4") {
                return FALSE;
            }
            break;
        case 'mastercard':
            if (strlen($ccnum) != 16 || !preg_match('/^5[1-5]/',
                $ccnum)) {
                return FALSE;
            }
            break;
        case 'amex':
            if (strlen($ccnum) != 15 || !preg_match('/^3[47]/',
                $ccnum)) {
                return FALSE;
            }
            break;
        case 'discover':
            if (strlen($ccnum) != 16 || substr($ccnum,0,4) == '6011'){
                return FALSE;
            }
            break;
        default:
            return FALSE;
    }
    // Starte MOD 10-Tests
    $dig = CharToArray($ccnum);
    $numdig = sizeof($dig);
    for ($i=($numdig-2), $j=0 ; $i>=0; $i-=2, $j++) {
        $dbl[$j] = $dig[$i] * 2;
    }
    $dblsz = sizeof($dbl);
    $validate = 0;
    for ($i=0; $i<$dblsz; $i++) {
        $add = CharToArray($dbl[$i]);
        for ($j=0; $j<sizeof($add); $j++) {
            $validate += $add[$j];
        }
        $add = '';
    }
}
```

Listing 8.4:
Prüfsummencheck
für gängige
Kreditkarten:
cc_check

V

S

I

1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E

8.1.6 Generierung einer WHERE-Bedingung

Die Funktion durchsucht eine Zeichenkette nach logischen Operatoren (and, or, not) und erstellt daraus den WHERE-Teil einer SQL-konformen Abfrage. Damit lässt sich beispielsweise eine Suchfunktion komfortabler gestalten.

```
<?php
function b_parse($str, $field) {
    if($str) {
        $quoted = explode( "\\\"", $str);
        for($i = 0; $i < count($quoted); $i++) {
            if($i == 0 && !$quoted[$i]) {
                // Anführungszeichen erzeugen, falls nicht vorhanden
                $begin = True;
                $i++;
            } /* end if */
            if($begin) {
                $words[] = $quoted[$i];
            } else {
                $phrase = explode( " ", $quoted[$i]);
                for($n = 0; $n < count($phrase); $n++) {
                    if($phrase[$n]) { $words[] = $phrase[$n]; }
                } /* end for */
            } /* end if */
            $begin = !$begin;
        } /* end for */
        for($i = 0; $i < count($words); $i++) {
            if($words[$i]) {
                if($words[$i] == "AND" || $words[$i] == "OR"
                    || $words[$i] == "NOT") {
                    if($words[$i] == "NOT") {
                        $i++;
                        if($sql_out) { $sql_out .= " AND "; }
                        $sql_out .= $field . " NOT LIKE '%" .
                            $words[$i] . "%'";
                    } elseif($i > 0) {
                        $sql_out .= " " . strtoupper($words[$i]) . " ";
                        $boolean = True;
                    }
                }
            } else {
                if($sql_out && !$boolean) {
                    $sql_out .= " OR ";
                }
                $sql_out .= $field . " LIKE '%" . $words[$i] . "%'";
                $boolean = FALSE;
            } /* end if */
        } /* end if */
    } /* end for */
} /* end if */
return $sql_out;
```

Listing 8.5:
Generierung einer
WHERE-Bedingung:
genwhere


```
}
?>
```

Wie es funktioniert Die Funktion setzt voraus, dass der Nutzer bei der Suchanfrage logische Ausdrücke verwendet. Diese können »and«, »or« oder »not« sein. Außerdem können Wörter zu Gruppen zusammengefasst werden. Die Wortgruppen werden zuerst zerlegt:

```
$quoted = explode( "\\\"", $str);
```

Die Schleife durchsucht nun die Zeichenkette nach alleinstehenden Wörtern.

```
for($i = 0; $i < count($quoted); $i++) {
```

Die Gruppen werden an den Anfang gestellt und übersprungen:

```
if($i == 0 && !$quoted[$i]) {
    $begin = True;
    $i++;
}
if($begin) { $words[] = $quoted[$i]; }
```

Alleinstehende Wörter werden durch Leerzeichen erkannt:

```
} else {
    $phrase = explode( " ", $quoted[$i]);
    for($n = 0; $n < count($phrase); $n++) {
        if($phrase[$n]) { $words[] = $phrase[$n]; }
    }
}
```

Die nächste Schleife durchläuft nun alle Wörter einzeln:

```
for($i = 0; $i < count($words); $i++) {
```

Wenn als alleinstehendes großgeschriebenes Wort ein logischer Operator erkannt wird, erfolgt eine Sonderbehandlung:

```
if($words[$i] == "AND"
|| $words[$i] == "OR"
|| $words[$i] == "NOT") {
```

Der Operator »NOT« wird in AND NOT LIKE %suchwort% umgewandelt, wobei *suchwort* das dem Operator nächstfolgende Wort ist:

```
if($words[$i] == "NOT") {
    $i++;
    if($sql_out) { $sql_out .= " AND "; }
    $sql_out .= $field . " NOT LIKE '%" .
        $words[$i] . "%'";
```

Ist der Operator nicht »not«, wird aus »and« AND und aus »or« OR gebildet und dem Ergebnis angehängt:

```
else if($i > 0) {
    $sql_out .= " ".strtoupper($words[$i])." ";
```

Wurden keine weiteren Operatoren gefunden, wird eine Oder-Verknüpfung angenommen und nach Wortfragmenten mit Hilfe von LIKE gesucht:

```
if($sql_out && !$boolean) {
    $sql_out .= " OR ";
    $sql_out .= $field." LIKE '%" . $words[$i] . "%'";
}
```

Das folgende Beispiel zeigt, wie Sie die SQL-Anweisung letztlich zusammensetzen können: **Anwendung**

```
if (isset($search)) {
    echo "SELECT * FROM address WHERE <br>" ;
    echo "      " . b_parse($search, "city") . "<br>";
    echo "      " . "OR" . "<br>";
    echo "      " . b_parse($search, "name");
}
```

Die Musterapplikation enthält ein Suchformular, dass entsprechende Anfragen erlaubt, aus denen die Bedingung erzeugt wird, wie die folgende Abbildung zeigt:

Abbildung 8.3:
Reaktion des Skripts

8.2 Gästebuch

Das Gästebuch ist eine typische Anwendung. Im letzten Kapitel wurde bereits ein Gästebuch zur Demonstration der Access-Verwendung gezeigt. Etwas anders aufgebaut ist die hier vorgestellte Variante, die wie alle Datenbankbeispiele dieses Kapitels auf MySQL aufbaut.

8.2.1 Datenbankstruktur

Das folgende Listing zeigt die Struktur der MySQL-Datenbank. Sie können dies als SQL-Anweisung ausführen oder die Informationen aus CREATE TABLE entnehmen und mit einem gängigen Werkzeug eingeben, beispielsweise mit PHPMYADMIN (siehe ➡ Abschnitt 7.6 *Administration mit phpMyAdmin* ab Seite 586).

```
# Tabellenstruktur für Tabelle 'guestbook'
CREATE TABLE guestbook (
    id int(8) DEFAULT '0' NOT NULL auto_increment,
    name varchar(30) DEFAULT '' NOT NULL,
    email varchar(30) DEFAULT '' NOT NULL,
    job varchar(30) DEFAULT '' NOT NULL,
    location varchar(30) DEFAULT '' NOT NULL,
    comments text DEFAULT '' NOT NULL,
```

Listing 8.6:
Tabellenstruktur für
das Gästebuch:
[myguestbook.sql](#)

```
url varchar(50),
PRIMARY KEY (id)
);
```

8.2.2 Der Code des Gästebuches

Das folgende Listing zeigt den Code auf einen Blick. Die wichtigen Stellen sind im Anschluss hervorgehoben.

Listing 8.7:
Gästebuch mit
MySQL-Datenbank:
myguestbook

```
<?php
// Datenbank verbinden
include("scripts/open.inc.php4");
// Hinzufügen von Einträgen - Formular
switch ($cmd) {
case 'add':
    echo <<<GUESTFORM
    <p>Bitte teilen Sie uns hier Ihre Meinung mit.</p>
    <p>
    <form name="guestbook" action="$PHP_SELF" method="post">
    <input type="Hidden" name="cmd" value="send"/>
    <table border="0">
    <tr><td>Ihr Name</td>
        <td><input type="text" name="name"/></td></tr>
    <tr><td>Ihre E-Mail Adresse</td>
        <td><input type="text" name="email"/></td></tr>
    <tr><td>Ihre Homepage</td>
        <td><input type="text" name="url"/></td></tr>
    <tr><td>Ihr Job</td>
        <td><input type="text" name="job"/></td></tr>
    <tr><td>Ihr Wohnort</td>
        <td><input type="text" name="location"/></td></tr>
    <tr><td>Kommentare</td>
        <td><textarea name="comments" cols="60" rows="6">
            </textarea></td>
    </tr>
    <tr>
        <td></td>
        <td>
            <input type="submit" value="Absenden"/>
            <input type="reset" value="Löschen"/>
        </td>
    </tr>
    </table>
    </form>
GUESTFORM;
    // Anzeige nach dem eigenen Eintrag
    break;
case 'view':
    echo '<h2>Anzeige der Einträge</h2>';
    // Datenbank abfragen
    $result = mysql_query("SELECT * FROM guestbook");
```

```

// Datensätze holen
while ($row = mysql_fetch_row($result)) {
    echo <<<ENTRY
    <HR>
    <b>Name:</b> $row[0]
    <br /><b>E-mail:</b><a href="mailto:$row[1]">$row[1]</a>
    <br /><b>Homepage:</b><a href="$row[2]">$row[2]</a>
    <br /><b>Job:</b> $row[3]
    <br /><b>Aus:</b> $row[4]
    <br /><b>Kommentar:</b>
    <br />$row[5]
ENTRY;
}
break;
case 'send':
    $comments = addslashes("$comments");
    $strQuery = "INSERT into guestbook (name, email, url, job,
                                location, comments) ";
    $strQuery .= "VALUES ('$name', '$email', '$url', '$job',
                        '$location', '$comments')";
    $result = @mysql_query($strQuery);
    if ($result) echo "<p>Danke! Wir werden Ihren Beitrag
                        ber&uuml;cksichtigen</p>";
    break;
default:
    // Zuletzt wird die Anzahl der Datensätze ermittelt
    // und zur Hauptseite zurückverwiesen
    $result = mysql_query("SELECT COUNT(*) FROM guestbook");
    $row = mysql_fetch_row($result);
    $num = $row[0];
    if ($num == "") {
        $entry = "Es sind zur Zeit keine Beitr&auuml;ge";
    } elseif ($num == "1") {
        $entry = "Es ist zur Zeit ein Beitrag";
    } else {
        $entry = "Es sind zur Zeit $num Beitr&auuml;ge";
    }
    echo <<<LINKS
    <p>Willkommen in unserem G&auuml;stebuch.<br>
    $entry im G&auuml;stebuch.<p>
    <a href="$PHP_SELF?cmd=add">Einen Beitrag
    hinzuf&uuml;gen</a><br />
    <a href="$PHP_SELF?cmd=view">Beitr&auuml;ge ansehen</a><br />
LINKS;
} /* end switch */
echo <<<ACCEPTED
    <p>
    <a href="$PHP_SELF">
    Zur&uuml;ck zur Startseite des G&auuml;stebuches</a>
    </p>

```

```
ACCEPTED;  
?>
```

8.2.3 Beschreibung

Wie es funktioniert Der Umgang mit der Datenbank selbst dürfte keine Probleme bereiten. Interessant ist die Verwendung der Parameter zur Steuerung. Immerhin sind alle Funktionen in einem einzigen Skript realisiert.

Die Links haben folgendes Aussehen:

```
<a href="$PHP_SELF?cmd=view">Betr&auml;ge ansehen</a>
```

Im Formular wird ein verstecktes Feld verwendet:

```
<input type="Hidden" name="cmd" value="send"/>
```

Sie werden dann im switch-Zweig durch Abfrage der Variablen *\$cmd* ausgewertet. Für jedes Kommando: View (Ansehen der Einträge), Send (Eintragen eines Beitrags) und Add (Formular zum Eintragen eines Beitrags) werden eigene HTML-Teile erzeugt. Dadurch wird das Gästebuch auf ein einziges Skript reduziert. Komplexere Ausbaustufen würden allerdings zu einem großen und schlecht zu pflegenden Skript führen. Das gezeigte Verfahren eignet sich eher für kleinere Projekte.

Beim Schreiben der Daten in die Datenbank ist lediglich zu beachten, dass Anführungszeichen markiert werden müssen:

```
$comments = addslashes("$comments");
```

Anschließend wird direkt das INSERT-Kommando für die Datenbank erzeugt. In der Praxis bietet es sich an, hier noch Prüfungen der Daten voranzustellen, damit keine leeren Einträge oder solche mit sinnlosen Informationen generiert werden.

8.3 Authentifizierung mit Datenbank

Die Authentifizierung von Nutzern wird häufig verlangt. Mit Einsatz einer Datenbank lässt sich das Problem besonders einfach lösen.

8.3.1 Datenbankstruktur

Das kleine Programm zeigt, wie die Authentifizierung nach Einträgen in einer MySQL-Datenbank erfolgen kann. Sie benötigen eine Datenbanktabelle *auth* mit folgender Struktur:

```
#
# Tabellenstruktur für Tabelle 'auth'
#
CREATE TABLE auth (
  id int(11) NOT NULL auto_increment,
  name varchar(20) NOT NULL,
  password varchar(20) NOT NULL,
  PRIMARY KEY (id),
  KEY name (name)
);
```

Listing 8.8: Struktur der Tabelle zur Authentifizierung: myauthentic.sql

8.3.2 Funktionsweise und Code

Die hier gezeigte Form setzt voraus, dass PHP als Modul im Apache läuft, denn nur dann stehen die Werte der Authentifizierung in den entsprechenden Variablen. Beachten Sie die HTTP-Header, die am Anfang des Skripts erzeugt werden.

```
<?php
function authenticate() {
  header('WWW-authenticate: basic realm=\"Photo Album\"');
  header('HTTP/1.0 401 Unauthorized');
  echo <<<INFO
  Sie benötigen einen gültigen Nutzernamen
  ...
  INFO;
  exit;
}
// Hauptprogramm
if(!isset($PHP_AUTH_USER)) {
  authenticate();
} else {
  include("open.inc.php");
  $id=strtolower($PHP_AUTH_USER);
  $query = mysql_query("SELECT * FROM users
                        WHERE name='$id'
                        AND password='$PHP_AUTH_PW'");
  if(!mysql_num_rows($query)) {
    authenticate();
  }
}
?>
```

Listing 8.9: Authentifizierung mit Datenbank-abfrage: myauthentic

Die Funktion nutzt die HTTP-Standardauthentifizierung. Dazu wird dem Browser eine entsprechende Aufforderung gesendet. **Wie es funktioniert**

```
Header("WWW-authenticate: basic realm=\"MyRealm\"");
Header("HTTP/1.0 401 Unauthorized");?>
```

Im Hauptprogramm wird die Variable \$PHP_AUTH_USER abgefragt, die nur gesetzt ist, wenn der Browser Nutzernamen und Kennwort abgefragt und übertragen hat:

```
if(!isset($PHP_AUTH_USER)) {
```

Dann wird die Tabelle *auth* geöffnet. Ändern Sie die Werte entsprechend Ihren Bedingungen in der Datei *OPEN.INC.PHP*.

Der Nutzernamen wird aus *\$PHP_AUTH_USER* und das Kennwort aus *\$PHP_AUTH_PW* geholt und die Spalten *id* (Name) und *password* werden abgefragt:

```
$id=strtolower($PHP_AUTH_USER);
$query = mysql_query("SELECT * FROM users
                      WHERE id='$id'
                      AND password='$PHP_AUTH_PW'");
```

Wurde der Nutzer nicht gefunden, wird die Abfrage des Kennwortes wiederholt:

```
if(!mysql_num_rows($query)) {
    authenticate();
```

8.4 Umfrage

Immer wieder nett und beliebt sind Umfragen, deren Ergebnisse üblicherweise erst ausgegeben werden, wenn der Nutzer sich beteiligt. Vor allem Fanseiten und private Homepages können mit einer solchen Anwendung ihre Attraktivität steigern.

8.4.1 Funktionsweise

Die hier vorgestellte Applikation Umfrage nutzt eine MySQL-Datenbank mit einer einzigen Tabelle. Neben dem unspektakulären HTML-Formular ist eine einfache Auswertung Bestandteil des Programms. Die Verwendung der Datenbankfunktionen ist hier nicht besonders umfangreich. Interessanter sind dagegen die SQL-Anweisungen – ein großer Teil der Auswertungen geschieht in der Datenbank selbst unter Zuhilfenahme der Aggregat-Funktionen.

8.4.2 Datenbankstruktur

Das folgende Listing zeigt die einzige Tabelle der Datenbank UMFRAGE.

Listing 8.10:
Tabellenstruktur für
die Umfrage:
umfrage.sql

```
#
# Tabellenstruktur für Tabelle 'umfrage'
#
CREATE TABLE umfrage (
  bundesland varchar(30),
  standort int(11),
  onlinezeit int(11),
  zugang int(11),
```

```

kosten int(11),
bestellt char(2),
summe double,
service int(11),
id int(11) NOT NULL auto_increment,
PRIMARY KEY (id)
);

```

8.4.3 Der Code der Umfrage

Das Programm besteht aus zwei Teilen – dem HTML-Formular UMFRAGE.PHP4 und der Auswertung AUSWERTUNG.PHP4. Sie finden beide Skripte im Verzeichnis ANWENDUNGEN/UMFRAGE auf der CD bzw. im entsprechenden Ordner der Website.

```

<html>
<head>
<title>Umfrage</title>
<link rel="stylesheet" type="text/css" href="umfrage.css">
</head>
<body bgcolor="#eeeeee">
<h1>Umfrage</h1>
<h2>Herzlich Willkommen zu unserer neuen Umfrage</h2>
<div class=text>
Um die Ergebnisse der Umfrage ansehen zu k&ouml;nnen, f&uuml;llen
Sie alle Fragen aus und klicken Sie auf
absenden.
<p>Die Umfrage ist selbstverst&auml;ndlich anonym.</p>
<form action="auswertung.php4" method="post">
<table bgColor="Silver" border=0 cellpadding=1 cellspacing=2
width=100%>
  <tr>
    <td bgColor="#eeeeee" colSpan=2><b>Zuerst Fragen nach der
Herkunft</b></td>
  </tr>
  <tr>
    <td bgColor=#ffe4b5>Aus welchem Bundesland kommen Sie?</td>
    <td bgColor=#ffe4b5><select name=bundesland>
      <option >Baden-W&uuml;rttemberg
      <option >Bayern
      <option selected>Berlin
      <option >Brandenburg
      <option >Bremen
      <option >Hamburg
      <option >Hessen
      <option >Mecklenburg-Vorpommern
      <option >Niedersachsen
      <option >Nordrhein-Westfalen
      <option >Rheinland-Pfalz
      <option >Saarland
      <option >Sachsen

```

Listing 8.11:
Formular zum
Absenden der
Umfrage: umfrage


```

        <option >Sachsen-Anhalt
        <option >Schleswig-Holstein
        <option >Th&uuml;ringen
    </select>
</tr>
<tr>
    <td bgColor="#ffe4b5">Von wo aus gehen Sie ins Internet?
    </td>
    <td bgColor="#ffe4b5">
        <input checked name=standort type=radio value=1> zu Hause
        <input name=standort type=radio value=2>im B&uuml;ro
        <input name=standort type=radio value=3>Uni/Schule
        <input name=standort type=radio value=3>
        zu Hause und B&uuml;ro</td>
</tr>
<tr>
    <td bgColor="#eeeeee" colSpan=2>
        <b>Fragen zur Benutzung des Internet</b>
    </td>
</tr>
<tr>
    <td bgColor="#ffe4b5">Wie viele Stunden pro Woche sind Sie
        online?</td>
    <td bgColor="#ffe4b5">
        <input name=onlinezeit type=text>Stunden durchschnittlich
    </td>
</tr>
<tr>
    <td bgColor="#ffe4b5">Welche Art Zugangstechnik haben Sie?
    </td>
    <td bgColor="#ffe4b5">
        <input name=zugang type=radio value=1> ISDN
        <input checked name=zugang type=radio value=2> Modem
        <input name=zugang type=radio value=3> LAN/Gateway</td>
</tr>
<tr>
    <td bgColor="#ffe4b5">Wie hoch sind ca. Ihre
        Online-Kosten?</td>
    <td bgColor="#ffe4b5"><input name=kosten type=text> &euro;
        (Provider+Telefon)</td>
</tr>
<tr>
    <td bgColor="#ffe4b5">Haben Sie schon mal was online
        gekauft?</td>
    <td bgColor="#ffe4b5"><input name=bestellt type=checkbox
        value=on>
        Ja, ich habe online bestellt und bezahlt</td>
</tr>
<tr>
    <td bgColor="#ffe4b5">Wie viel haben Sie im letzten Jahr
        online gekauft?
    <td bgColor="#ffe4b5"><input name=summe type=text> &euro;

```

[illegible]

Die Auswertung schreibt die Daten des Formulars in die Datenbank und fragt die aktualisierten Daten sofort ab.

```
<?php
if (!isset($kosten)) $kosten = 0;
if (!isset($summe)) $summe = 0;
if ($bundesland==' ' or $standort==' ' or $onlinezeit==' ') {
    header("Location: $HTTP_REFERER");
    exit;
}
include ("../scripts/open.inc.php4");
$query = "INSERT INTO umfrage (bundesland, standort, onlinezeit,
zugang, kosten, bestellt, summe, service)";
$query .= "VALUES ('$bundesland', $standort, $onlinezeit, $zugang,
$kosten, '$bestellt', $summe, $service)";
mysql_query($query, $conn);

?>
<html>
<head>
    <title>Auswertung</title>
    <link rel="stylesheet" type="text/css" href="umfrage.css">
</head>
<body bgcolor="#eeeeee">
<h1>Auswertung der Umfrage</h1>
<div class=text>
Die Ergebnisse sind eine Zusammenfassung der Angaben aller
bisherigen Teilnehmer.
```

*Listing 8.12:
Auswertung der
Umfrage und
Berechnung der
Ergebnisse (Skript
auswertung). Die
SQL-Abfragen sind
hervorgehoben*

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

```

<p>
<table bgColor="Silver" border=0 cellPadding=1 cellSpacing=2
width=100%>
  <tr><th>Fragestellung</th><th>Ergebnis</th></tr>
  <tr>
    <td valign="top" bgcolor=#ffe4b5>
      Aus welchen Bundesl&auml;ndern <br />
      kamen wieviele Besucher?</td>
    <td bgcolor=#ffe4b5 valign="top">
      <table border=1 cellPadding=2 cellSpacing=0
        bordercolor="white" width="100%">
        <?php
          $query = "SELECT COUNT(bundesland), bundesland
                     FROM umfrage GROUP BY bundesland";
          $rs = mysql_query($query, $conn);
          while ($row = mysql_fetch_row($rs)) {
            echo <<<ROW
            <tr>
              <td>$row[1]</td>
              <td>$row[0]</td>
            </tr>
          ROW;
          }
        ?>
      </table>
    </td>
  </tr>
  <tr>
    <td valign="top" bgColor=#ffe4b5>
      Wo steht der Computer?</td>
    <td bgcolor=#ffe4b5 valign="top">
      <?php
        $query = "SELECT COUNT(standort), standort
                   FROM umfrage GROUP BY standort";
        $rs = mysql_query($query, $conn);
        while ($row = mysql_fetch_row($rs)) {
          switch ($row[1]) {
            case 1:
              $zh = (int) $row[0];
              break;
            case 2:
              $ib = (int) $row[0];
              break;
            case 3:
              $us = (int) $row[0];
              break;
          }
        }
        $summe = $zh + $ib + $us;
        $p_zh = sprintf("%.2f", $zh / $summe * 100);
        $p_ib = sprintf("%.2f", $ib / $summe * 100);
        $p_us = sprintf("%.2f", $us / $summe * 100);
      </?php>
    </td>
  </tr>
</table>

```

```

        echo <<<INF01
        $zh Computer zu Hause = $p_zh % <br />
        $ib Computer im Büro = $p_ib % <br />
        $us Computer in UNI/Schule = $p_us %
INF01;
        ?>
        </td>
    </tr>
    <tr>
        <td valign="top" bgColor=#ffe4b5>
            Welche Zugangstechnik wird benutzt?
        </td>
        <td bgColor=#ffe4b5 valign="top">
            <?php
                $query = "SELECT COUNT(zugang), zugang
                           FROM umfrage GROUP BY zugang";
                $rs = mysql_query($query, $conn);
                while ($row = mysql_fetch_row($rs)) {
                    switch ($row[1]) {
                        case 1:
                            $id = (int) $row[0];
                            break;
                        case 2:
                            $mo = (int) $row[0];
                            break;
                        case 3:
                            $ln = (int) $row[0];
                    }
                }
                $summe = $zh + $ib + $us;
                $p_id = sprintf("%.2f", $id / $summe * 100);
                $p_mo = sprintf("%.2f", $mo / $summe * 100);
                $p_ln = sprintf("%.2f", $ln / $summe * 100);
                echo <<<INF02
                $id ISDN = $p_id % <br />
                $mo Modem = $p_mo % <br />
                $ln LAN/Gateway = $p_ln %
INF02;
        ?>
        </td>
    </tr>
    <tr>
        <td bgColor=#ffe4b5 nowrap>
            Wie viele Stunden sind Sie im Schnitt online?</td>
        <td bgColor=#ffe4b5>
            <?php
                $query = "SELECT AVG(onlinezeit) FROM umfrage";
                $rs = mysql_query($query, $conn);
                $row = mysql_fetch_row($rs);
                printf('Im Durchschnitt wird %.2f Stunden
                        pro Monat gesurft', $row[0]);

```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

```

?>
</td>
</tr>
<tr>
<td bgColor=#ffe4b5>Wie hoch sind die Kosten?</td>
<td bgColor=#ffe4b5>
<?php
$query = "SELECT AVG(kosten) FROM umfrage";
$rs = mysql_query($query, $conn);
$row = mysql_fetch_row($rs);
printf("Im Durchschnitt kostet der Online-Zugang %0.2f
&euro; pro Monat.", $row[0]);
?>
</td>
</tr>
<tr>
<td bgColor=#ffe4b5>
Wer hat schon mal online gekauft?
Und welchen Werte hatte die Bestellung?
</td>
<td bgColor=#ffe4b5>
<?php
$query = "SELECT COUNT(*), COUNT(bestellt),
          AVG(summe) FROM umfrage
          WHERE bestellt='on'";
$rs = mysql_query($query, $conn);
$row = mysql_fetch_row($rs);
if ($row[0] == 0) {
    echo 'Niemand hat Online bestellt';
} else {
    $bst = sprintf("%0.2f", $row[1] / $row[0] * 100);
    $sum = sprintf("%0.2f", $row[2]);
    echo <<<INF03
        $bst % der Nutzer haben schon online bestellt.
        Dabei wurden im Durchschnitt $sum &euro; ausgegeben.
INF03;
    }
?>
</TD>
</tr>
<tr>
<td valign="top" bgColor=#ffe4b5>
Wie gut fanden Sie unsere Website?</td>
<td bgColor=#ffe4b5 valign="top">
<?php
$query = "SELECT COUNT(service), service,
          AVG(service) AS average
          FROM umfrage GROUP BY service";
$rs = mysql_query($query, $conn);
while ($row = mysql_fetch_row($rs))

```

```

    {
        switch ($row[1]) {
            case 1:
                $z1 = $row[0];
                break;
            case 2:
                $z2 = $row[0];
                break;
            case 3:
                $z3 = $row[0];
                break;
            case 4:
                $z4 = $row[0];
                break;
            case 5:
                $z5 = $row[0];
                break;
        }
    }
    $summe = $z1 + $z2 + $z3 + $z4 + $z5;
    $p_z1 = sprintf("%.2f", $z1 / $summe * 100);
    $p_z2 = sprintf("%.2f", $z2 / $summe * 100);
    $p_z3 = sprintf("%.2f", $z3 / $summe * 100);
    $p_z4 = sprintf("%.2f", $z4 / $summe * 100);
    $p_z5 = sprintf("%.2f", $z5 / $summe * 100);
    $a = ($z1)+($z2*2)+($z3*3)+($z4*4)+($z5*5);
    $avg = sprintf("%.0f", $a/$summe);
    echo <<<INF04
    1: $z1 Teilnehmer, entspricht: $p_z1 %<br />
    2: $z2 Teilnehmer, entspricht: $p_z2 %<br />
    3: $z3 Teilnehmer, entspricht: $p_z3 %<br />
    4: $z4 Teilnehmer, entspricht: $p_z4 %<br />
    5: $z5 Teilnehmer, entspricht: $p_z5 %<br />
    <p>
    Die Durchschnittsnote ist: $avg
INF04;
?>
    </td>
</tr>
</table>
</div>
</body>
</html>

```

8.4.4 Erläuterung der Arbeitsweise

Das Formular UMFRAGE.PHP4 besteht aus reinem HTML und soll hier nicht weiter erklärt werden. Bei der Auswertung (das ist das Skript AUSWERTUNG.PHP4) werden zwei Aktionen ausgeführt:

- Schreiben der neuen Daten in die Datenbank

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

- Auslesen der Datenbank und Auswerten

Die Formularvariablen werden in den entsprechenden PHP-Variablen erwartet. Wenn bestimmte Felder nicht ausgefüllt wurden, werden die Variablen 0 gesetzt oder es erfolgt eine Umleitung auf das Formular. Der Nutzer kann damit fehlerhaft ausgefüllte Formulare nicht absenden.

```
if (!isset($kosten)) $kosten = 0;
if (!isset($summe)) $summe = 0;
if ($bundesland==" " or $standort==" " or $onlinezeit==" ") {
    header("Location: $HTTP_REFERER");
    exit;
```

Im nächsten Schritt wird eine Verbindung zur Datenbank eröffnet und die Daten des Formulars werden in die Tabelle eingefügt.

Die eigentliche Arbeit erledigen mehrere SELECT-Anweisungen, die neben der Abfrage der Daten auch gleich die Auswertung vornehmen. Die »Intelligenz« des Programms steckt also in der Datenbank.

Die erste Abfrage ermittelt die Anzahl der Nutzer pro Bundesland. Feld 0 enthält die Anzahl (berechnet mit COUNT) und Feld 1 den Namen des Bundeslandes selbst. Damit die Länderliste eindeutig ist, wird sie mit GROUP BY gruppiert:

```
$query = "SELECT COUNT(bundesland), bundesland
FROM umfrage GROUP BY bundesland";
```

Die Auswertung der Nutzungsplätze erfolgt in gleicher Weise:

```
$query = "SELECT COUNT(standort) , standort
FROM umfrage GROUP BY standort";
```

Allerdings wird hier noch eine prozentuale Berechnung angeschlossen. Dazu werden die Werte einzeln selektiert (switch) und dann mit der Summe berechnet.

```
while ($row = mysql_fetch_row($rs)) {
    switch ($row[1]) {
        case 1:
            $zh = $row[0];
            break;
        case 2:
            $ib = $row[0];
            break;
        case 3:
            $us = $row[0];
            break;
    }
}
$summe = $zh + $ib + $us;
$p_zh = sprintf("%.2f", $zh / $summe * 100);
```

```
$p_ib = sprintf("%.2f", $ib / $summe * 100);
$p_us = sprintf("%.2f", $us / $summe * 100);
```

Durch die GROUP BY-Bedingung tritt übrigens jede Variante nur einmal auf, die Schleife wird tatsächlich nur drei- beziehungsweise fünfmal durchlaufen.

Die durchschnittlichen Online-Kosten und die beim Surfen verbrachte Zeit ist ebenso ein Wert, der direkt in der Datenbank über alle Felder mit der Funktion AVG berechnet wird:

```
$query = "SELECT AVG(onlinezeit) FROM umfrage";
$query = "SELECT AVG(kosten) FROM umfrage";
```

Die Anzahl der Käufer im Internet und der durchschnittliche Wert eines Kaufs wird mit der folgenden Anweisung ermittelt. Zum Einsatz kommen die Funktionen COUNT und AVG, die Abfrage wird auf Datensätze eingeschränkt, bei denen das Feld *bestellt* gleich »on« ist:

```
$query = "SELECT COUNT(*), COUNT(bestellt),
          AVG(summe) FROM umfrage
          WHERE bestellt='on'";
```

Analog der Auswertung der Nutzungsplätze wird mit der Note für den Service umgegangen:

```
$query = "SELECT COUNT(service), service
          FROM umfrage GROUP BY service";
```

Die Schleife weist den Variablen nun wieder die Werte zu, bei fünf möglichen Antworten wird die Schleife nur fünfmal durchlaufen.

```
while ($row = mysql_fetch_row($rs)) {
    switch ($row[1]) {
        case 1:
            $z1 = $row[0];
            break;
        case 2:
            $z2 = $row[0];
            break;
        case 3:
            $z3 = $row[0];
            break;
        case 4:
            $z4 = $row[0];
            break;
        case 5:
            $z5 = $row[0];
    }
}
```

Jetzt ist die Anzahl der Wertungen pro Note bekannt. Im letzten Schritt werden die prozentualen Anteile berechnet und die Durchschnittsnote bestimmt.


```
$summe = $z1 + $z2 + $z3 + $z4 + $z5;  
$p_z1 = sprintf("%.2f", $z1 / $summe * 100);  
$p_z2 = sprintf("%.2f", $z2 / $summe * 100);  
$p_z3 = sprintf("%.2f", $z3 / $summe * 100);  
$p_z4 = sprintf("%.2f", $z4 / $summe * 100);  
$p_z5 = sprintf("%.2f", $z5 / $summe * 100);  
$a = ($z1)+($z2*2)+($z3*3)+($z4*4)+($z5*5);
```

Am Ende des Abschnitts finden Sie die Abbildung des Umfrage-Formulars und einer Auswertung. In der Gestaltung der Umfrage sind Sie natürlich völlig frei. Beim praktischen Einsatz sollte bedacht werden, dass Sicherheitsmaßnahmen zum Schutz vor Missbrauch notwendig sind. Dazu gehören:

- Kontrolle einer Mehrfachteilnahme

Verhinderung, dass Nutzer mehrfach teilnehmen, beispielsweise durch IP-Sperren und Cookies (➡ Abschnitt 4.3 *Cookies* ab Seite 324).

- Prüfung der Eingabedaten

Prüfung auf sinnvolle Werte und Zusammenhang zwischen Werten, beispielsweise durch reguläre Ausdrücke (➡ Abschnitt 3.2.12 *Reguläre Ausdrücke* ab Seite 217).

- Manipulationsversuche erkennen

Verhinderung von Manipulationsversuchen durch gefakte²⁶ Formulare, beispielsweise durch Auswertung des Referers (➡ Abschnitt 4.7.1 *Die Umgebungsvariablen des Webserver* ab Seite 379).

Übungsaufgaben

Versuchen Sie, das Skript mit den entsprechenden Methoden zu erweitern und professioneller zu gestalten. Der Einsatz ist vielfältig möglich, denn Umfragen aller Art sind beliebt und weit verbreitet im Internet. Man kann beispielsweise die Anzeige der Ergebnisse zwingend an eine sinnvolle Eingabe koppeln (die gezeigte Version ist leicht zu übergeben), um einen Anreiz zur Teilnahme zu schaffen.

²⁶ Als »fake« (engl. Fälschung) wird ein Betrugsversuch bezeichnet.

Umfrage

Herzlich Willkommen zu unserer neuen Umfrage

Um die Ergebnisse der Umfrage ansehen zu können, füllen Sie alle Fragen aus und klicken Sie auf absenden.

Die Umfrage ist selbstverständlich anonym.

Zuerst Fragen nach der Herkunft

Aus welchem Bundesland kommen Sie?

Von wo aus gehen Sie ins Internet? ☐ zu Hause ☐ im Büro ☐ Uni/Schule ☒ zu Hause und Büro

Fragen zur Benutzung des Internet

Wie viele Stunden pro Woche sind Sie online? Stunden durchschnittlich

Welche Art Zugangstechnik haben Sie? ☒ ISDN ☐ Modem ☐ LAN/Gateway

Wie hoch sind ca. Ihre Online-Kosten? € (Provider+Telefon)

Haben Sie schon mal was online gekauft? ☒ Ja, ich habe online bestellt und bezahlt

Wie viel haben Sie im letzten Jahr online gekauft? € (ca. Gesamtwert)

Wie zufrieden waren Sie mit dem Service? ☒ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 (Schulnoten)

Abbildung 8.4:
Das Formular der
Umfrage

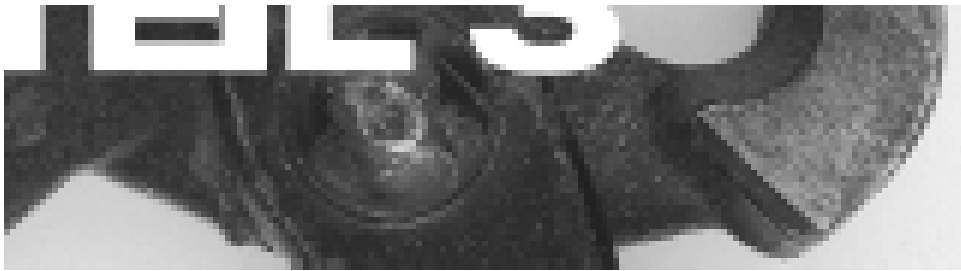
Auswertung der Umfrage

Die Ergebnisse sind eine Zusammenfassung der Angaben aller bisherigen Teilnehmer.

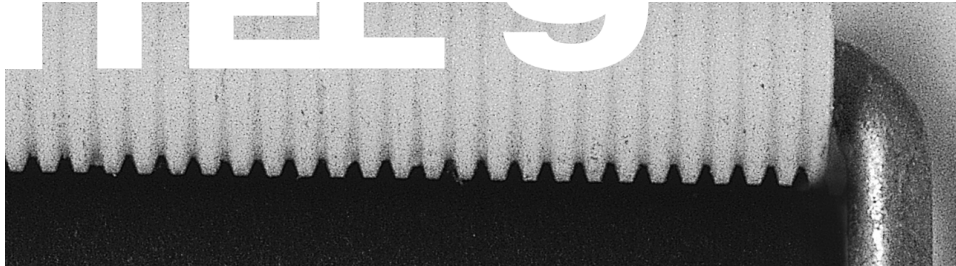
Fragestellung	Ergebnis								
Aus welchen Bundesländern kamen wieviele Besucher?	<table border="1"> <tr><td>Berlin</td><td>4</td></tr> <tr><td>Bremen</td><td>13</td></tr> <tr><td>Hamburg</td><td>4</td></tr> <tr><td>Niedersachsen</td><td>2</td></tr> </table>	Berlin	4	Bremen	13	Hamburg	4	Niedersachsen	2
Berlin	4								
Bremen	13								
Hamburg	4								
Niedersachsen	2								
Wo steht der Computer?	8 Computer zu Hause = 34.78 % 11 Computer im Büro = 47.83 % 4 Computer in Uni/Schule = 17.39 %								
Welche Zugangstechnik wird benutzt?	18 ISDN = 78.26 % 5 Modem = 21.74 % LAN/Gateway = 0.00 %								
Wie viele Stunden sind Sie im Schnitt online?	Im Durchschnitt wird 25.83 Stunden pro Monat gesurft								
Wie hoch sind die Kosten?	Im Durchschnitt kostet der Online-Zugang 165.09 € pro Monat.								
Wer hat schon mal online gekauft? Und welchen Werte hatte die Bestellung?	100.00 % der Nutzer haben schon online bestellt. Dabei wurden im Durchschnitt 339.48 € ausgegeben.								
Wie gut fanden Sie unsere Website?	1: 4 Teilnehmer, entspricht: 17.39 % 2: 15 Teilnehmer, entspricht: 65.22 % 3: 1 Teilnehmer, entspricht: 4.35 % 4: 3 Teilnehmer, entspricht: 13.04 % 5: Teilnehmer, entspricht: 0.00 % Die Durchschnittsnote ist: 2								

Abbildung 8.5:
Die Auswertung der
Umfrage

TEIL III



PHP professionell
programmieren



Fortgeschrittene Programmierung

Dieses Kapitel wendet sich an fortgeschrittene Programmierer, die sich größeren und anspruchsvolleren Aufgaben widmen möchten. Behandelt werden seltenere Funktionen und vor allem solche, die ein tiefer gehendes Wissen in Bezug auf die Arbeitsweise eines Webservers erfordern. Für Anfänger ist in jedem Kapitel ein theoretischer Abschnitt zu finden, sodass Sie schnell zu den Profis aufschließen können.

Netzwerkprogrammierung (Seite 631)

E-Mail-Programmierung (Seite 635)

Portable Document Format – PDF (Seite 654)

Ausgabesteuerung (Seite 666)

Reguläre Ausdrücke (Seite 668)

Extensible Markup Language – XML (Seite 697)

Wireless Markup Language – WML (Seite 724)

Web Distributed Data Exchange – WDDX (Seite 733)

9.1 Netzwerkprogrammierung

Unter Netzwerkprogrammierung wird hier die Programmierung von Funktionen verstanden, die sich auf Protokolle des Internet beziehen, also FTP, HTTP usw.

9.1.1 Netzwerkfunktionen

Im vorangegangenen Abschnitt wurden die theoretischen Grundlagen der Netzwerkprogrammierung gezeigt. Haben Sie nun entsprechende Systeme verfügbar – Webserver, FTP-Server oder Nameserver –, können Sie mit PHP darauf zugreifen. Die dazu benötigten Funktionen werden in diesem Abschnitt vorgestellt.

Die Socket-Funktionen

In Kapitel 4 wurden Sie bereits mit den Funktionen `fsocketopen` und `pfsocketopen` konfrontiert. Dort ging es um die Behandlung von Dateien – und die können bekanntlich auch auf einem Webserver liegen. Bei der Programmierung von Anwendungen, die gezielt auf den Server zugreifen, sind jedoch eine ganze Reihe weiterer Funktionen notwendig.

Zur Verbindung mit einem Server spezifizieren Sie den Domain-Namen oder die IP-Adresse und den Port: **fsocketopen()**

```
$fp = fsocketopen("hostname.tld", "80");
```

Mit dem so erzeugten Handle `$fp` können weitere Operationen mit entsprechenden Funktionen ausgeführt werden. So lassen sich mit den Dateifunktionen viele Probleme schnell lösen.

Typisch ist die Möglichkeit, eine Datei vom Server herunterzuladen. Das ist an sich sehr einfach. Mancher Browser bietet aber nicht den Dialog zum Download, sondern versucht die Datei anzuzeigen. Das folgende Skript verhindert dies: **Download steuern**

```
<?php
$file = @fopen($fileName, 'r');
header("Content-Type: $application");
header("Content-Disposition: attachment; filename=$fileName");
header("Content-Description: PHP4 Generated Data");
header("Pragma: no-cache");
header("Expires: 0");
fpassthru($file);
fclose($file);
?>
```

*Listing 9.1:
network_download
(aufgerufen über
network_html)
initiiert ein
Download per Skript*

header()

Das Programm verwendet im Wesentlichen die bereits vorgestellten HTTP-Header, um die entsprechende Reaktion im Browser auszulösen. Die Datei selbst wird mit den Dateifunktionen geladen.

Das nächste Beispiel verwendet denselben Mechanismus. Diesmal werden Bilder zum Download angeboten. Wenn der Nutzer auf ein Bild klickt, wird dieses nicht erneut angezeigt, sondern geladen.

*Listing 9.2:
network pictures:
Download von
Bildern*

```
<?php
if($command == 1) {
    header("Content-Disposition: attachment; filename=$filename");
    header("Content-Type: application/photoshop");
    header("Content-Length: ".filesize("$filename"));
    header("Pragma: no-cache");
    header("Expires: 0");
    $fp=fopen("$filename","r");
    print fread($fp, filesize("$filename"));
    fclose($fp);
    exit();
} /* end if */
?>

<?php
if($command == 0) {
    $filename = 'download.gif';
    echo <<<HTML
    <body bgcolor="#eeeeee">
    Klicken Sie auf das Bild, um es zu laden:
    <p>
    <a href="$PHP_SELF?command=1&filename=$filename">
    
    </a>
    HTML;
} /* end if */
?>
```

Die Abfrage eines Whois-Servers offenbart vielfältige Informationen über eine Domain. Sie können einen solchen Abfrageservice leicht anbieten, wenn Sie das folgende Skript einsetzen:

*Listing 9.3:
Komfortable Abfrage
mehrerer Whois-
Server:
network whois*

```
<?php
function lookup($lookup, $server){
    $errno = $errstr = "";
    $fp = fsockopen($server, 43, &$errno, &$errstr, 30);
    if (!$fp){
        printf("Error: %s (%s)", $errstr, $errno);
        $data = 0;
    } else {
        $lookup .= "\n";
        fputs($fp, $lookup);
        $data = fread($fp, 16384 );
        fclose($fp);
    }
}
```

```

    return $data;
}
?>
<form method="POST" action="<?php echo $PHP_SELF; ?>">
Whois-Server:
<select name="server">
    <option value="whois.ripe.net">Ripe</option>
    <option value="whois.denic.de" selected>DE</option>
    <option value="whois.nic.ch">CH</option>
    <option value="whois.networksolutions.com">COM/EDU</option>
</select>
Domain: <input type="text" name="lookup"/>
<input type="submit" value="Abfrage starten"/>
</form>
<p />
<?php
if(isset($lookup)) {
    $data = lookup($lookup, $server);
    echo "Die Abfrage dauert ein paar Sekunden, bitte warten Sie";
    printf("<pre>\n%s\n</pre>\n", $data);
}
?>

```

Die Abbildung auf der nächsten Seite zeigt, wie das Skript sich bei einer konkreten Anfrage verhält.

Die entscheidende Zeile im Skript ist der Punkt, wo das Handle für die Verbindung erstellt wird:

```
$fp = fsockopen($server, 43, &$errno, &$errstr, 30);
```

43 ist der Port für die Whois-Dienste, 30 steht für die Zeit, die für die Anfrage zur Verfügung steht. Die Abfrage selbst muss mit einem Zeilenende abgeschlossen werden, was mit der folgenden Zeile erzeugt wird:

```
$lookup .= "\n";
```

Der Abruf der Daten erfolgt mit der folgenden Sequenz:

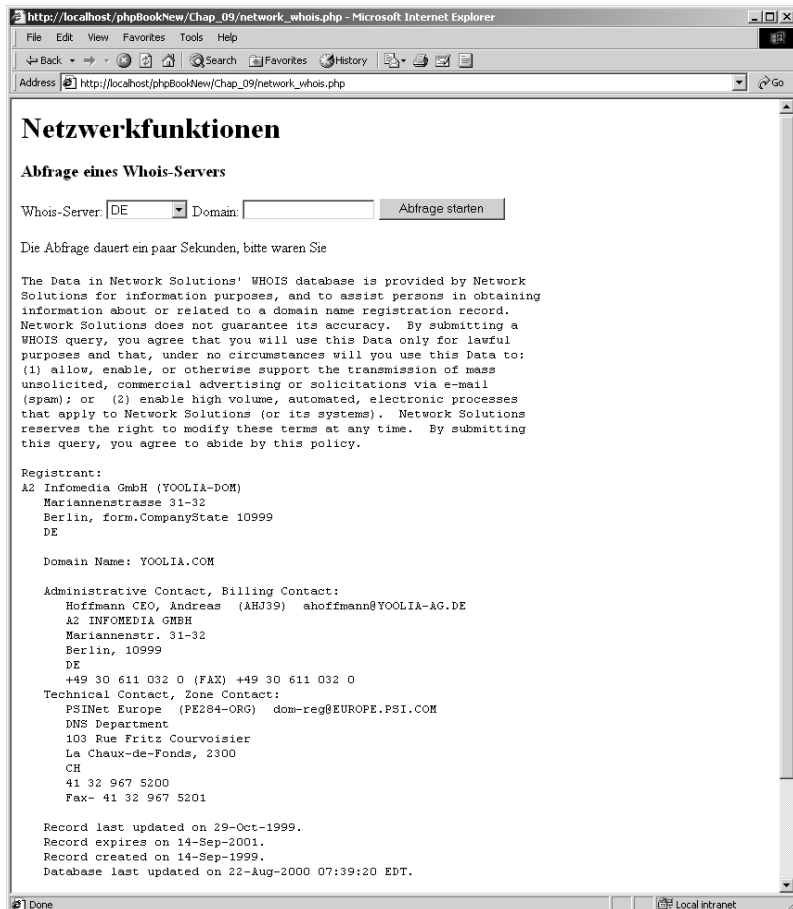
```

fputs($fp, $lookup);
$data = fread( $fp, 16384);
fclose($fp);

```

Zuerst wird die Anfrage mit fputs gesendet. Dann wartet fread auf die Antwort.

Abbildung 9.1:
Das Skript aus
Listing 9.3 in Aktion



DNS-Funktionen

Funktionsübersicht Die folgende Übersicht zeigt alle Funktionen, die im Zusammenhang mit dem Domain Name System genutzt werden können. Darüber hinaus können Sie Verbindungen zu Nameservern mit fsockopen herstellen.

- `gethostbyaddr`

Ermittelt den Namen eines Hosts anhand einer IP-Adresse.

- `gethostbyname`

Die Funktion ermittelt die IP-Adresse zu einem Hostnamen.

- `gethostbyname1`

Hiermit wird eine Liste von korrespondierenden IP-Adressen zu einem gegebenen Namen erstellt.

- `checkdnsrr`

Die Funktion prüft DNS-Einträge zu einem Namen oder einer IP-Adresse. Als Eintrag darf A, MX, NS, SOA, PTR, CNAME oder ANY angegeben werden. Der Standardwert ist MX.

- `getmxrr`

Diese Funktion prüft alle MX-Einträge und gibt diese in einem Array zurück.

Ob ein Eintrag vorhanden ist, prüft die Funktion `checkdnsrr`:

```
<?php
if (checkdnsrr('domain.com' 'A')) {
    echo "domain.com hat einen A-Eintrag";
}
?>
```

`checkdnsrr()`

Listing 9.4:
`checkdnsrr`:
Prüfen eines DNS-Eintrags

Die Funktion wird von der aktuellen Windows-Distribution nicht unterstützt, Sie müssen zum Testen Zugriff auf einen Linux-Server haben.

Nicht für Windows!

Die IP-Adresse zu einem Namen kann man, wenn ein Nameserver dies zu beantworten in der Lage ist, mit der Funktion `gethostbyname` ermitteln. Die Anwendung zeigt das folgendes Skript. Denken Sie daran, dass für eine eindeutige Auflösung die Angabe des Servernamens (normalerweise »www«) erforderlich ist.

`gethostbyname()`

```
<?php
if (isset($domain)) {
    echo "Die IP-Adresse für <b>$domain</b> ist: ";
    echo gethostbyname($domain);
}
?>
<form action="$PHP_SELF" method="post">
    Domain: <input type="Text" name="domain" value="www."/>
           <input type="Submit" value="IP-Adresse ermitteln"/>
</form>
```

Listing 9.5:
`gethostbyname`: IP-Adresse zu einer Domain ermitteln

Die IP-Adresse für **www.spiegel.de** ist: 194.64.249.245

Domain:

Abbildung 9.2:
Ausgabe des Skripts
aus Listing 9.5

Die DNS-Funktionen sind nicht geeignet, einen Nameserver aufzubauen oder zu simulieren. Die Anwendung setzt dagegen den Zugriff auf einen Nameserver voraus.



9.2 E-Mail-Programmierung

Der eigentliche Vorgang des Absendens ist auf nur eine Funktion beschränkt. Umfangreicher dagegen ist die Formatierung der E-Mail-

Nachricht. Wenn Sie derartige Funktionen benötigen, sollten Sie unbedingt die theoretischen Grundlagen lesen.

9.2.1 POP3 und SMTP programmieren

Als Beispiel für den Umgang mit den E-Mail-Protokollen wird hier ein Skript gezeigt, das POP3 »spricht«. Die Technik lässt sich leicht auf SMTP übertragen.

Abfrage eines POP3-Servers

Das folgende Listing zeigt eine Klasse, die mehrere Methoden zum Senden von POP3-Kommandos enthält. In der richtigen Reihenfolge abgerufen, lassen sich damit POP3-Server abfragen, die Nachrichten lesen und löschen.

Anwendung

Vielleicht rufen Sie Ihre E-Mail mit einem Clientprogramm vom PC aus ab. Dann mag ein derartiges Programm nicht sinnvoll erscheinen. es gibt aber vielfältige Anwendungsmöglichkeiten:

- Realisieren Sie ein Spam-Filter, indem Sie online bestimmte Kriterien erfassen und die Nachrichten daraufhin überprüfen. Sie können dann selektiv das DELE-Kommando senden, um unerwünschte E-Mails zu löschen.
- Löschen Sie zu große E-Mails vor dem Laden. Eine Größenanzeige ist zwar nicht direkt möglich. Die Anbindung Ihres Servers beim Provider ist aber sicher besser als der Client zu Hause. Dann sind 5 Megabyte schnell geladen und ebenso schnell gelöscht.
- Lesen Sie Ihre E-Mails per Webschnittstelle, auch wenn Ihr Provider dies nicht unterstützt. So ist auch der Zugriff aus dem Urlaubsort kein Problem. Belassen Sie die Nachrichten auf dem Server, damit Sie diese mit Ihrem gewohnten Client abrufen können, wenn Sie von der Reise zurückkehren.

Das folgende Listing zeigt die Klasse und ein Abrufbeispiel. Ersetzen Sie den Namen des Servers sowie Benutzername und Kennwort durch die Angaben, die zu Ihrer Mailbox passen.

*Listing 9.6:
Beispiel email_pop3:
Klasse zur Abfrage
eines POP3-Servers
und Ausgabe der
Nachrichten im
Browser (fette Wörter
sind systemabhängig)*

```
<?php
class pop3 {

    var $strStatus;
    var $pop3;
    var $arrMessages;
    var $arrMessage;

    function pop3 () {
        $this->strStatus = array();
```

```
$this->pop3 = 0;
$arrMessages = $arrMessage = '';
}

function open($strServer, $intPort = 110) {
    $this->pop3 = fsockopen($strServer, $intPort);
    if (!is_resource($this->pop3)) return FALSE;
    $line = fgets($this->pop3, 1024);
    return $this->getresult($line);
}

function user($strUser) {
    fputs($this->pop3, "USER $strUser\r\n");
    $line = fgets($this->pop3, 1024);
    return $this->getresult($line);
}

function pass($strPass) {
    fputs($this->pop3, "PASS $strPass\r\n");
    $line = fgets($this->pop3, 1024);
    return $this->getresult($line);
}

function mess() {
    fputs($this->pop3, "LIST\r\n");
    $line = fgets($this->pop3, 1024);
    if ($this->getresult($line)) {
        while(substr($line=fgets($this->pop3, 1024),0,1) != '.')
        {
            $this->arrMessages[] = $line;
        }
        return TRUE;
    } else {
        return FALSE;
    }
}

function retr($strMessage) {
    list($intMessage) = explode(' ', $strMessage);
    fputs($this->pop3, "RETR $intMessage\r\n");
    $line = fgets($this->pop3, 1024);
    if ($this->getresult($line)) {
        unset($this->arrMessage);
        while(substr($line=fgets($this->pop3, 1024), 0 ,1)!='.')
        {
            $this->arrMessage[] = $line;
        }
        return TRUE;
    } else {
        return FALSE;
    }
}
```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

```
}

function dele($intMessage) {
    fputs($pop3, "DELE $intMessage\r\n");
    $line = fgets($pop3, 1024);
    return TRUE;
}

function quit() {
    fputs($this->pop3, "QUIT\r\n");
    $line = fgets($this->pop3, 1024);
    return $this->getresult($line);
}

function getresult($line)
{
    $this->strStatus = substr($line, 0, 1024);
    if (substr($this->strStatus, 0, 1) != '+') {
        return FALSE;
    } else {
        return TRUE;
    }
}

function showresult()
{
    echo '<code style="color:red">';
    echo $this->strStatus;
    echo '</code><br>';
}

} /* end class pop3 */

$mybox = new pop3;

// stellen Sie hier die Parameter eines Ihnen bekannten
// POP3-Kontos ein
$popserver = 'popserver.domain.de';
$username = 'username';
$password = 'password';
echo 'Verbindung: ';
$error = FALSE;
while (TRUE)
{
    if (!$mybox->open($popserver)) {
        echo "[FEHLER] Verbindung nicht möglich<br>\n";
        $mybox->showresult();
        $error = TRUE;
        break;
    } else {
        echo "erfolgreich<br>\n";
        $mybox->showresult();
    }
}
```

```

    }
    echo 'Benutzername: ';
    if (!$mybox->user($username)) {
        echo "[FEHLER] Benutzername nicht erkannt!<br>\n";
        $mybox->showresult();
        $error = TRUE;
        break;
    } else {
        echo "erfolgreich angemeldet<br>\n";
        $mybox->showresult();
    }
    echo 'Kennwort: ';
    if (!$mybox->pass($password)) {
        echo "[FEHLER] Kennwort falsch!<br>\n";
        $mybox->showresult();
        $error = TRUE;
        break;
    } else {
        echo "akzeptiert<br>\n";
        $mybox->showresult();
    }
    echo 'Nachrichten: ';
    if (!$mybox->mess()) {
        echo "[FEHLER] Abruf fehlgeschlagen!<br>\n";
        $error = TRUE;
        break;
    } else {
        echo 'erfolgreich <b>' . count($mybox->arrMessages) . "</b>
gelesen<br>\n";
        $mybox->showresult();
    }
    break;
} /* end while */
echo '<hr noshade size="2">';
if (!$error and is_array($mybox->arrMessages)) {
    foreach($mybox->arrMessages as $strMessage) {
        echo "<b>Nachricht [$strMessage]</b>:<br>\n";
        if ($mybox->retr($strMessage)) {
            foreach($mybox->arrMessage as $strData) {
                echo "$strData<br>";
            }
        } else {
            $mybox->showresult();
        }
        echo '<hr noshade size="2">';
    } /* end foreach */
}
}

```

Die Funktionen in Listing 9.6 stellen nicht mehr als ein Framework für **Wie es funktioniert** eigene Versuche dar. Die verfügbaren Funktionen sind aber eine gute

Abbildung des POP3-Protokolls. Die Eigenschaften der Klasse haben folgende Bedeutung:

- *\$strStatus*
Letzte Statuszeile des POP3-Servers
- *\$pop3*
Dateihandle der Verbindung
- *\$arrMessages*
Array mit den Nummern der Nachrichten
- *\$arrMessage*
Array der Daten der aktuellen Nachricht

Die Funktionen unterscheiden sich im Detail nur wenig. Zur Eröffnung der Verbindung wird die Funktion `fsockopen` verwendet:

```
function open($strServer, $intPort = 110) {  
    $this->pop3 = fsockopen($strServer, $intPort);  
    if (!is_resource($this->pop3)) return FALSE;
```

Die Antwort auf diese Abfrage wird analysiert und zurückgegeben:

```
    $line = fgets($this->pop3, 1024);  
    return $this->getresult($line);  
}
```

Das Lesen der Nachricht erfolgt durch Abrufen der gesendeten Zeilen mit `fgets`:

```
function mess() {  
    fputs($this->pop3, "LIST\r\n");  
    $line = fgets($this->pop3, 1024);  
    if ($this->getresult($line)) {  
        unset($this->arrMessages);  
        while(substr($line = fgets($this->pop3, 1024),0,1) != '.'){  
            $this->arrMessages[] = $line;  
        }  
        return TRUE;  
    } else {  
        return FALSE;  
    }  
}
```

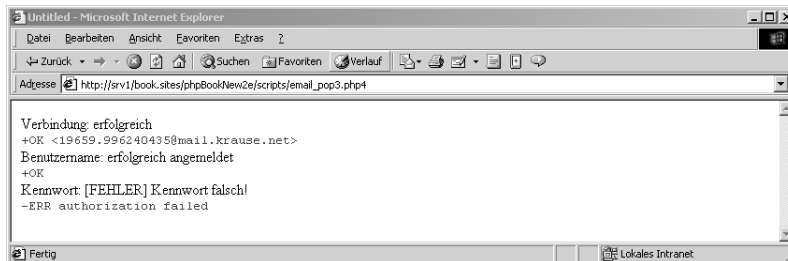


Abbildung 9.3:
Ausgabe bei
fehlerhafter
Anmeldung

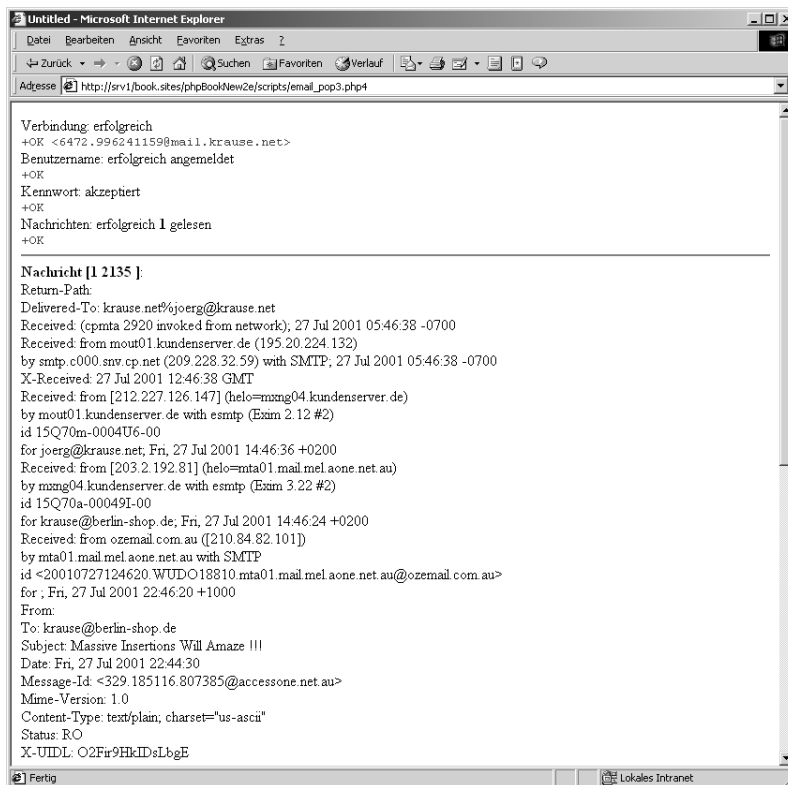


Abbildung 9.4:
Ausgabe bei
erfolgreicher Abfrage

Die Verwendung der Klasse erfolgt durch Aufruf der Methoden in der vom Protokoll bestimmten Reihenfolge: Öffnen der Verbindung, Übertragung von Benutzername und Kennwort und Abruf der Nachrichten. Die while-Schleife wird hier trickreich eingesetzt. Tatsächlich soll diese Schleife nie mehr als einmal durchlaufen werden. Dafür sorgt die break-Anweisung am Ende. Die einzelnen Kommandos in der Schleife werden nacheinander verarbeitet. Tritt dabei ein Fehler auf, wird die Schleife abgebrochen und damit die Ausführung der übrigen Kommandos verhindert.

Verwendung

```
$mybox = new pop3;
$popserver = 'popserver.domain.de';
$username = 'username';
```

```

$password = 'password';
while (TRUE) {
    if (!$mybox->open($popserver)) {
        echo "[FEHLER] Verbindung nicht möglich<br>\n";
        $mybox->showresult();
        $error = TRUE;
        break;
    } else {
        echo "erfolgreich<br>\n";
        $mybox->showresult();
    }
}

```

Die Nachrichten selbst werden in Arrays übermittelt, sodass die Auswertung und Trennung der Kopfzeilen leicht erfolgen kann.

```

if ($mybox->retr($strMessage)) {
    foreach($mybox->arrMessage as $strData) {
        echo "$strData<br>";
    }
} else {
    $mybox->showresult();
}

```

Die Ausgabe der Kopfzeilen kann zum Kennlernen des Aufbaus einer E-Mail aber auch sehr hilfreich sein.



Es bleibt der Fantasie des Lesers überlassen, hier anzusetzen und verschiedene Dekodierungen der Mailstandards einzubauen. Ebenso könnte die rudimentäre Fehlerverwaltung verbessert werden.

SMTP

Die Programmierung von SMTP-Funktionen ist ebenso wie schon bei POP3 gezeigt relativ einfach. Basis sollte eine Funktion sein, die Kommandos aller Art sendet und empfängt. Die bestehende TCP/IP-Verbindung wird dabei einfach mit übergeben. Das folgende Beispiel ist alleine nicht lauffähig, bietet aber das nötige Rüstzeug für eigene Versuche.

Listing 9.7:
email_smtp:
Funktions-
Framework für die
Bedienung eines
SMTP-Servers

```

<?php
class smtp {

    var $strStatus;
    var $strSMTP;

    function smtp() {
        $this->strStatus = '';
        $this->strSMTP = 0;
    }

    function open($server, $port = 25) {
        $this->strSMTP = fsockopen($server, $port);
    }
}

```

```
    if ($this->strSMTP < 0) return FALSE;
    $line = fgets($this->strSMTP, 1024);
    return $this->strSMTP;
}

function helo($strHost) {
    fputs($this->strSMTP, "HELO $strHost\r\n");
    $line = fgets($this->strSMTP, 1024);
    return $this->getresult($line, 2);
}

function from($from) {
    fputs($this->strSMTP, "MAIL FROM: <$from>\r\n");
    $line = fgets($this->strSMTP, 1024);
    return $this->getresult($line, 2);
}

function to($to) {
    fputs($this->strSMTP, "RCPT TO: <$to>\r\n");
    $line = fgets($this->strSMTP, 1024);
    return $this->getresult($line, 2);
}

function data($subject, $data) {
    fputs($this->strSMTP, "DATA\r\n");
    while ($line = fgets($this->strSMTP, 1024))
        if ($this->getresult($line, 3)) break;
    $this->showresult();
}

if ($this->getresult($line, 2)) {
    fputs($this->strSMTP, "Subject: $subject\r\n");
    fputs($this->strSMTP, "$data\r\n\r\n");
    fputs($this->strSMTP, ".\r\n");
    $line = fgets($this->strSMTP, 1024);
    return $this->getresult($line, 2);
} else {
    return FALSE;
}

function quit() {
    fputs($this->strSMTP, "QUIT\r\n");
    $line = fgets($this->strSMTP, 1024);
    return $this->getresult($line, 2);
}

function getresult($line, $code)
{
    $this->strStatus = substr($line, 0, 1024);
    if (substr($this->strStatus, 0, 1) != $code) {
        return FALSE;
    }
}
```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

```
    } else {  
        return TRUE;  
    }  
}  
  
function showresult()  
{  
    echo '<code style="color:red">';  
    echo $this->strStatus;  
    echo '</code><br>';  
}  
  
} /* end class pop3 */  
  
$mybox = new smtp;  
// Ändern Sie diese Werte für Ihren SMTP-Server  
$smtpserver = 'smtp.domain.de';  
$pcname = 'myPC';  
$fromname = 'meine@mailadresse.de';  
$toname = 'joerg@krause.net';  
$error = FALSE;  
while (TRUE) {  
    echo 'Verbindung: ';  
    if (!$mybox->open($smtpserver)) {  
        echo "[FEHLER] Verbindung nicht möglich<br>\n";  
        $mybox->showresult();  
        $error = TRUE;  
        break;  
    } else {  
        echo "erfolgreich<br>\n";  
        $mybox->showresult();  
    }  
    echo 'Anmeldung: ';  
    if (!$mybox->hello($pcname)) {  
        echo "[FEHLER] HELO fehlgeschlagen!<br>\n";  
        $mybox->showresult();  
        $error = TRUE;  
        break;  
    } else {  
        echo "erfolgreich angemeldet<br>\n";  
        $mybox->showresult();  
    }  
    echo 'Absender: ';  
    if (!$mybox->from($fromname)) {  
        echo "[FEHLER] Kommando nicht erkannt!<br>\n";  
        $mybox->showresult();  
        $error = TRUE;  
        break;  
    } else {  
        echo "akzeptiert<br>\n";  
        $mybox->showresult();  
    }  
}
```

```

    }
    echo 'Empfänger: ';
    if (!$mybox->to($toname)) {
        echo "[FEHLER] Kommando nicht erkannt!<br>\n";
        $error = TRUE;
        break;
    } else {
        echo "akzeptiert<br>\n";
        $mybox->showresult();
    }
    break;
} /* end while */

echo 'Daten: ';
$data = <<<SMTPMAIL
Dieser Text wurde mit einer eigenen SMTP-Klasse gesendet

SMTPMAIL;
if (!$mybox->data('Testmail per SMTP', $data)) {
    echo "[FEHLER] Kommando nicht erkannt!<br>\n";
    $mybox->showresult();
    exit;
} else {
    echo "akzeptiert<br>\n";
    $mybox->showresult();
}
?>

```

Im Prinzip ähnelt der Aufbau der zuvor beschriebenen Klasse, die einen POP3-Server abfragt. Das Senden der Daten dürfte den einzigen Unterschied betreffen, da hier mehrere Zeilen nacheinander verarbeitet werden müssen.

Zuerst wird das Kommando DATA gesendet:

```
fputs($this->strSMTP, "DATA\r\n");
```

Der Server antwortet normalerweise mit mehreren die Übertragung betreffenden Hinweisen, die alle mit dem Statuscode 250 beginnen. Die Aufforderung zum Senden trägt dagegen den Code 354. Entsprechend werden solange Statuszeilen abgefragt, bis die Codegruppe 3 erreicht wird.

```

while ($line = fgets($this->strSMTP, 1024)) {
    if ($this->getresult($line, 3)) break;
}
if ($this->getresult($line, 3)) {

```

Dann folgen die Header der E-Mail, die Sie nach Bedarf um alles ergänzen können, wie in ➡ Abschnitt *Header der E-Mail-Nachrichten* ab Seite 100 beschrieben.

Wie es funktioniert

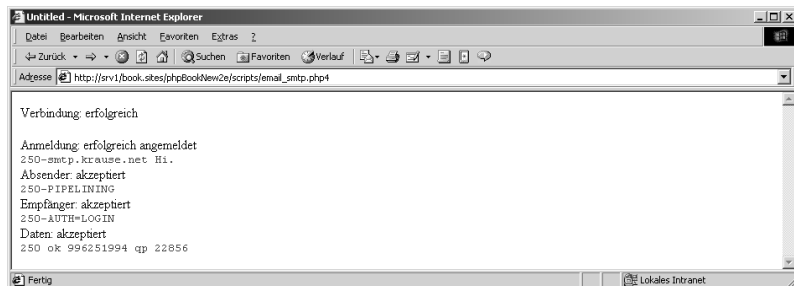
V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

```
fputs($this->strSMTP, "Subject: $subject\r\n");
```

Der Datenblock endet mit zwei Leerzeilen und einem alleinstehenden Punkt am Anfang einer Zeile.

```
fputs($this->strSMTP, "$data\r\n\r\n");
fputs($this->strSMTP, ".\r\n");
$line = fgets($this->strSMTP, 1024);
}
```

Abbildung 9.5:
Ausgabe, wenn das
Skript erfolgreich
verarbeitet wurde



Diskussion

Manche Server verlangen eine Authentifizierung mit AUTH, andere registrieren eine vorangehende erfolgreiche POP3-Authentifizierung. Diese können Sie leicht mit der in Listing 9.6 gezeigten Klasse starten. Freie Relays, die keine Authentifizierung mehr erfordern, sind sehr selten zu finden. Manchmal genügt es auch, zuvor einen anderen POP3-Client zu starten, wenn das Skript lokal getestet wird. In jedem Fall ist natürlich eine Verbindung zum Internet erforderlich.

IMAP-Funktionen

Wenn Sie sich die Funktionsübersicht in PHP anschauen, fallen viele IMAP-Funktionen auf. Einige dieser Funktionen lassen sich für viele E-Mail-Aufgaben einsetzen, egal ob im Zusammenhang mit POP3, SMTP oder auch IMAP. Prinzipiell eignen sie sich aber für den Umgang mit IMAP-Servern. Das IMAP-Protokoll war ursprünglich als Weiterentwicklung von POP3 entwickelt worden, konnte sich aber nicht in diesem Maße durchsetzen.

Ein Beispiel für den Einsatz der IMAP-Funktionen finden Sie in Listing 9.8. Dort werden sie zur Programmierung des Zugriffs auf einen Newsserver eingesetzt.

9.2.2 E-Mail unter Windows NT/2000

Die PHP-Dokumentation und viele Quellen im Internet gehen davon aus, dass das Programm *sendmail* installiert ist. Dies ist normalerweise unter Unix der Fall.



Windows NT hat standardmäßig kein SMTP-Mail-Programm. Die Serverversion des Option Packs enthält ein rudimentäres SMTP-

Programm, nicht jedoch die Workstation-Version. Windows 2000 liefert einen SMTP-Server mit, sowohl in der Server- als auch der Professional-Version. In der Konfigurationsdatei `PHP.INI` können Sie die Mailfunktion so konfigurieren, dass der SMTP-Server verwendet wird:

```
[mail function]
SMTP          = 192.168.100.12
sendmail_from = krause@comzept-gmbh.de
```

Der Eintrag `SMTP` bestimmt die Adresse oder IP-Nummer, `sendmail_from` den Standardabsender (der SMTP-Server benötigt diese Angabe).

Blat

BLAT ist eine Public Domain-Software, die unter Windows NT ein Konsolenwerkzeug zum Versenden von E-Mails zur Verfügung stellt. Damit BLAT arbeiten kann, muss eine Verbindung zu einem SMTP-Server bestehen.



Kopieren Sie die Datei `BLAT.EXE` nach `\WINNT\SYSTEM32` oder in ein anderes Verzeichnis, das sich im Pfad befindet. **Installation**

Starten Sie das Programm am Prompt mit:

```
C:>Blat -install domain.de mail@domain.de
```

BLAT beherrscht Base64 und UUencode. E-Mails werden direkt an einen SMTP-Server versendet. Verfügen Sie über einen Relay, können diese Werte in der Registrierung fest eingestellt werden. **Funktionen**

Blat verwendet folgende Syntax:

Parameter

```
Blat <filename> -t <recipient> [schalter]
Blat -install <server addr> <sender's addr> [-q]
Blat -h [-q]
```

Die Parameter haben folgende Bedeutung:

- Setzen den Standard-SMTP-Server und Absender.

```
-install <server addr> <sender's addr>
```

- Datei mit der Nachricht

```
<filename>
```

- Empfänger der Nachricht.

```
<recipient>
```

Folgende Schalter können optional eingesetzt werden:

- Kommaseparierte Liste der Empfänger

```
-t <recipient>
```


- Betreffzeile

```
-s <subj>
```

- Absenderadresse

```
-f <sender>
```

- 'From:'-Adresse

```
-i <addr>
```

- Cc-Liste

```
-c <recipient>
```

- BCC-Liste

```
-b <recipient>
```

- MIME Quoted-Printable Content-Transfer-Encoding

```
-mime
```

- Dieser Schalter unterdrückt alle Ausgaben

```
-q
```

- Anderer SMTP-Server

```
-server <addr>
```

- Hostname

```
-hostname <hst>
```

- UUEncoded verwenden

```
-uuencode
```

- Base64 versenden

```
-base64
```

Folgende Beispiele zeigen mögliche Aufrufe:

- Setzt Host und User-ID:

```
Blat -install smtphost.bar.com foo@bar.com  
Blat -install smtphost.bar.com foo
```

- Setzt nur den Host:

```
Blat -install smtphost.bar.com
```

- Sendet die Datei MYFILE.TXT mit der Betreffzeile »Eine Datei für Clemens« an *foo@bar.com*:

```
Blat myfile.txt -s "Eine Datei für Clemens" -t foo@bar.com
```



Beispiel

- Sendet die Datei »myfile.txt« mit der Betreffzeile »Eine Datei für Clemens« an *foo@bar.com* und unterdrückt die Ausgaben. -f überschreibt den voreingestellten Absender:

```
Blat myfile.txt -s "Eine Datei für Clemens" -t foo@bar.com -q
Blat myfile.txt -s "Eine Datei für Haide" -t fee@fi.com
-f foo@bar.com
```

- -i ersetzt die "From:"-Adresse, lässt aber "Reply-To:" und "Sender:" unverändert:

```
Blat myfile.txt -s "Eine Datei für Clemens" -t foo@bar.com
-i "andere@absender.de"
```

- Sendet eine Kopie (»Cc«) an *clemens@krause.net* und *haide.fk@berlin-shop.de*:

```
Blat myfile.txt -s "animals" -t fee@fi.com
-c "clemens@krause.net,haide.fk@berlin-shop.de"
```

- Sendet die Binärdatei BLAT17I.ZIP im Base 64-Format:

```
Blat BLAT17I.ZIP -s "Datei für Clemens" -t foo@bar.com -base64
```

- Sendet die Binärdatei BLAT17I.ZIP im UUEncode-Format:

```
Blat BLAT17I.ZIP -s "Datei für Clemens" -t foo@bar.com
-uuencode
```

- Sendet eine Nachricht über den SMTP-Server *smtp.domain.com* auf Port 6000:

```
Blat myfile.txt -t fee@fi.com -server smtp.domain.com:6000
```

- Teilt dem SMTP-Programm mit, dass der Host »friend« heißt:

```
Blat myfile.txt -t fee@fi.com -hostname friend
```

9.2.3 E-Mail aus Skripten versenden

Mit einer einzigen Funktion ist es möglich, eine E-Mail zu versenden. Voraussetzung ist ein lokal installierter SMTP-Server, was zumindest unter Linux standardmäßig der Fall sein dürfte. Alternativ bietet sich die in Listing 9.7 gezeigte SMTP-Klasse an, mit der auch exotische oder entfernte Mailserver problemlos angesprochen werden können.

Die Funktion mail

Mit `mail` versenden Sie eine E-Mail. Die Funktion gibt `FALSE` zurück, `mail()` wenn der Aufruf misslang. Die Syntax ist einfach und kann dem folgenden Beispiel entnommen werden:

```
mail($to, $subject, $message, $header);
```

Der erste Parameter bestimmt die Zieladresse, der zweite den Inhalt der Betreffzeile und der dritte die eigentliche Botschaft. Optional können Sie weitere Kopfzeilen (Header) mit dem vierten Parameter hinzufügen.

Optionale Header einstellen

Die Konstruktion der Header wurde in ➤ Abschnitt *Header der E-Mail-Nachrichten* ab Seite 100 beschrieben. Wenn Sie beispielsweise den Absender einstellen möchten, erzeugen Sie die Variable *\$header* wie folgt:

```
$header = "From: \"Joerg Krause\" <joerg@Krause.net>\n";
```

Vergessen Sie nicht das Zeilenendezeichen »\n«. Sie können damit auch mehrere Header kombinieren:

```
$header = "From: \"Joerg Krause\" <joerg@Krause.net>\n";
$header .= "X-Mailer: PHP-Version " . PHP_VERSION . "\n";
```

Die Nachricht selbst erstellen Sie mit den bereits im einführenden Teil enthaltenen Informationen so, wie der Empfänger es erwartet. Zeilenumbrüche bei reinen Textnachrichten werden mit »\n« erzeugt. Wenn Sie HTML versenden möchten, eignet sich die Nutzung des Puffers, wie in ➤ Abschnitt 9.4 *Ausgabesteuerung* ab Seite 666 beschrieben. Informationen über den Aufbau von HTML-E-Mails und weitere theoretische Grundlagen finden Sie in ➤ Abschnitt 2.2.4 *Grundlagen der E-Mail-Programmierung* ab Seite 89.

9.2.4 NNTP – für Newsserver programmieren

Das NNTP-Protokoll dient dem Zugriff auf Newsserver des Usenet. Der Standardport ist Port 119. Prinzipiell läuft die Kommunikation jedoch wie bei einer E-Mail ab, sodass sich der Einsatz der E-Mail-Funktionen für die Programmierung eines Newsdienstes anbietet. Im folgenden Beispiel wird ein Newsserver abgefragt und der Inhalt einer Nachrichtengruppe im Browser angezeigt. Den Server selbst können Sie auf jeder Plattform realisieren, freie Newsserver finden Sie für Linux und unter Windows NT/2000 in der Serverversion.

Code eines Newsabrufers

Das folgende Listing zeigt, wie mit den Netzwerkfunktionen auf einem Newsserver zugegriffen wird. Verwendet werden hier die imap-Funktionen. Das entsprechende Modul muss also vorhanden bzw. die passende DLL in der Datei PHP.INI aktiviert sein.

Listing 9.8:
Abfrage eines NNTP-
Servers: nntp_imap

```
<?php
class nntp {
    var $nntp;
```

```

function listm() {
    $num = imap_num_msg($this->nntp);
    echo <<<NNTPHEAD
        $num Nachrichten in php.dev gefunden, zeige 10 an<br>\n
        <table>
            <tr>
                <th align=left>Nr.</th>
                <th>Größe</th>
                <th align=left>Sender</th>
                <th align=left>Diskussion</th>
            </tr>
NNTPHEAD;
        for($i = $num - 10; $i <= $num; $i++) {
            $header = imap_header($this->nntp, $i);
            if(!$header->Size) continue;
            $from = $header->from;
            $addr = $from[0]->mailbox."@".$from[0]->host;
            $name = $from[0]->personal ? $from[0]->personal : $addr;
            echo '<tr><td>';
            echo "<a href=\"\$PHP_SELF?msg=$i\">$i</a></td>";
            echo "<td>".$header->Size."</td>";
            echo "<td><a href=\"mailto: $addr\">$name</a></td>\n";
            echo "<td>".$header->subject."</td>\n";
            echo "</tr>\n";
        }
        echo "</table>\n";
    }

    function message($msg) {
        $header = imap_header($this->nntp, $msg);
        $from = $header->from;
        $addr = $from[0]->mailbox."@".$from[0]->host;
        $name = $from[0]->personal;
        $date = date("d.M.Y H:i:s", $header->udate + 4 * 3600);
        $data = htmlspecialchars(imap_body($this->nntp,$msg));
        echo <<<NNTPMESSAGE
            <table border=1>\n
            <tr>
                <th align=left>Von: </th>
                <td>$name &lt;
                    <a href="mailto: $addr">$addr</a>&gt;
                </td>
            </tr>
            <tr>
                <th align=left>Diskussion:</th>
                <td>$header->Subject </td>\n
            <tr>
                <th align=left>Date: </th>
                <td>$date EDT</td></tr>
            <tr>
                <td colspan=2><pre>$data</pre></td>

```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

```

        </tr>
    </table>
    NNTPMESSAGE;
    }

    function nntp($server, $group) {
        $this->nntp = imap_open("{'.$server.'/nntp:119}$group",
                                "", "", OP_ANONYMOUS);
    }

    function close() {
        imap_close($this->nntp);
    }

} /* end class nntp */

$server = 'news.php.net';
$group  = 'php.dev';
$mybox = new nntp($server, $group);
$mybox->listm();
if (isset($msg)) {
    $mybox->message($msg);
}
$mybox->close();
?>

```

Wie es funktioniert Die Klasse enthält vier Methoden: Eine zum Öffnen der Verbindung – eine Verbindung zum Internet wird vorausgesetzt, eine zum Abfragen der Header der Nachrichten in der Gruppe, eine weitere zum Abruf einer bestimmten Nachricht und eine zum Schließen der Verbindung.

Newsserver und Gruppe werden mit dem Konstruktor der Klasse definiert:

```

$server = 'news.php.net';
$group  = 'php.dev';
$mybox = new nntp($server, $group);

```

imap_open() Dieser realisiert die Verbindung mit der Funktion `imap_open`:

```

$this->nntp = imap_open("{'.$server.'/nntp:119}$group",
                        "", "", OP_ANONYMOUS);

```

Beachten Sie die Syntax der Funktion: {server/protokoll:port}gruppe.

imap_num_msg() Die Anzahl der Nachrichten wird mit

```

$num = imap_num_msg($this->nntp);

```

abgefragt.

imap_header() Die Funktion `imap_header` gibt ein Objekt zurück, das die Informationen der Nachrichten als Eigenschaften enthält.

```
$header = imap_header($this->nntp, $msg);
```

Die Abfrage der Eigenschaften kann unmittelbar folgen, die folgenden Zeilen zeigen die Kopfdaten an:

```
if(!$header->Size) continue;
$from = $header->from;
$addr = $from[0]->mailbox."@".$from[0]->host;
$name = $from[0]->personal ? $from[0]->personal : $addr;
```

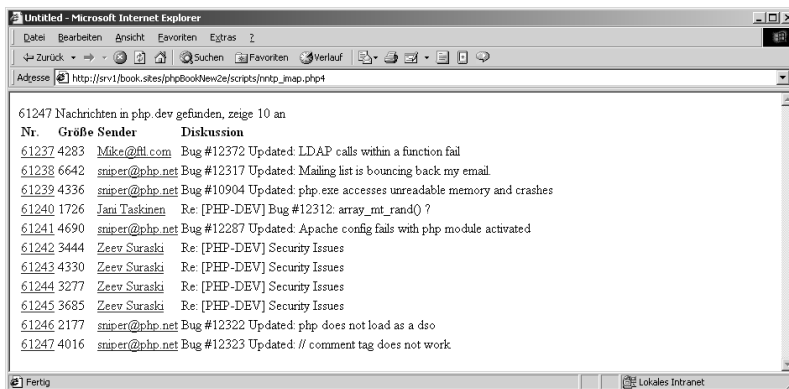
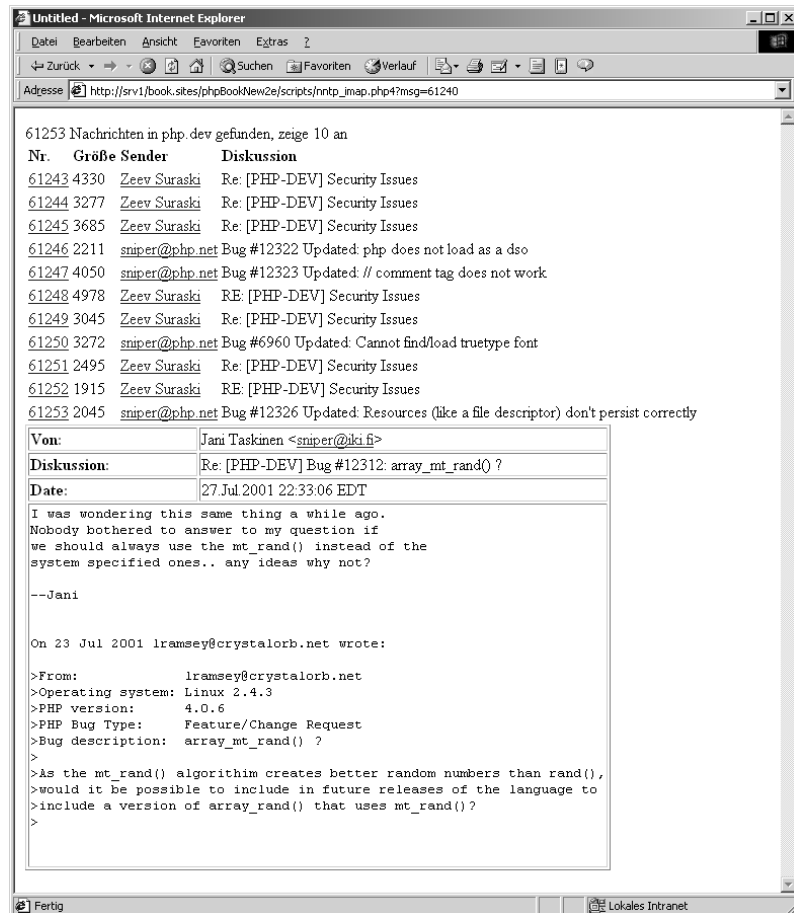


Abbildung 9.6:
Ausgabe der
Kopfzeilen eines
Newservers

Etwas umfangreicher ist die Abfrage bei der Darstellung einer einzelnen Nachricht:

```
$from = $header->from;
$addr = $from[0]->mailbox."@".$from[0]->host;
$name = $from[0]->personal;
$date = date("d.M.Y H:i:s", $header->update + 4 * 3600);
$data = htmlspecialchars(imap_body($this->nntp, $msg));
```

Abbildung 9.7:
Anzeige einer
bestimmten
Nachricht



9.3 Portable Document Format – PDF

Über Adobe PDF

Das Adobe Portable Document Format (PDF) ist ein offener Standard für die elektronische Dokumentenverteilung. PDF ist ein universelles Dateiformat, das alle Schriften, Formatierungen, Farben und Grafiken eines beliebigen Quelldokuments beibehält, unabhängig von dem Programm und dem Betriebssystem, mit dem es erstellt wurde. PDF-Dateien sind kompakt und können von jedermann gemeinsam genutzt, betrachtet und exakt ausgedruckt werden. Dies erfordert das kostenlos bei Adobe erhältliche Programm »Acrobat Reader«.

9.3.1 Grundlagen PDF

PDF ist ein Format für elektronische Dokumentenverteilung, weil es die üblichen Probleme der gemeinsamen Arbeit mit Dateien überwindet.

- Empfänger können Dateien nicht öffnen, weil sie nicht die Programme haben, mit denen die Dokumente erstellt wurden.
- Formatierungen, Schriften und Grafiken gehen aufgrund von inkompatiblen Betriebssystemen, Programmen und Versionen verloren.
- Dokumente werden aufgrund von Software- und Druckerbeschränkungen falsch ausgedruckt.
- Jeder kann überall eine PDF-Datei öffnen. Alles, was dazu benötigt wird, ist der kostenlose Acrobat Reader.
- PDF-Dateien werden immer exakt so angezeigt, wie sie erstellt wurden, unabhängig von Schriften, Software und Betriebssystemen.
- PDF-Dateien lassen sich auf allen Druckern immer korrekt ausgeben.
- PDF-Dateien können beliebig publiziert und verteilt werden: gedruckt, als E-Mail-Anlage, auf Firmen-Servern, auf Web-Sites oder CD-ROM.
- Die kompakten PDF-Dateien sind kleiner als ihre Quelldateien und lassen sich Seite für Seite herunterladen, damit sie im Web schnell angezeigt werden können.

Probleme mit elektronischen Dokumenten

Vorteile von PDF

Mit der Generierung von PDF-Dokumenten per PHP sind Sie unabhängig von der sonst notwendigen (nicht kostenlosen) Adobe Software, speziell Adobe Acrobat und Adobe Distiller. Allerdings ist die *pdflib* nur für private Zwecke kostenfrei. Bei der kommerziellen Nutzung fallen erhebliche Lizenzgebühren an. Deshalb ist diese Erweiterung bei Providern nur selten zu finden.

Nachteile der pdflib

9.3.2 Einführung in die pdflib

Die PDF-Funktionen sind Bestandteil der *pdflib* von Thomas Merz. Wenn sie Ihrer PHP-Distribution nicht beilieg, können Sie die neueste Version unter <http://www.pdflib.com> finden. Dort bekommen Sie auch eine aktuelle Dokumentation. Zum Verständnis ist jedoch etwas Grundwissen nützlich.

Außerdem sollten Sie wissen, dass PDF intern eine Einheit als ein 72tel Zoll annimmt.

PDF-Funktionen

Eine vollständige Liste aller PDF-Funktionen und deren Parameter finden Sie in der Referenz. Diese Auflistung hier enthält nur die in der aktuellsten Version verfügbaren Funktionen, nicht jedoch die aus Kompatibilitätsgründen zu älteren Versionen noch existierenden Varianten. Sie sollten sich bei der Entwicklung neuer Skripte deshalb unbedingt auf die folgenden Funktionen beschränken:

Funktionsliste

- `pdf_set_info`
Die Funktion setzt das Info-Feld der Dokumenteninformation.
- `pdf_open`
Öffnet ein neues PDF-Dokument und gibt ein Handle darauf zurück, das von anderen Funktionen verwendet wird.
- `pdf_close`
Schließt ein PDF-Dokument und gibt das Handle und Ressourcen frei.
- `pdf_begin_page`
Beginnt eine neue PDF-Seite.
- `pdf_end_page`
Beendet eine PDF-Seite.
- `pdf_show`
Die Funktion schreibt einen Text an die aktuelle Position.
- `pdf_show_boxed`
Schreibt einen Text in ein unsichtbares Rechteck.
- `pdf_show_xy`
Schreibt einen Text an die angegebene Position.
- `pdf_set_font`
Wählt einen Zeichensatz und dessen Größe aus.
- `pdf_set_parameter`, `pdf_get_parameter`
Die Funktion setzt und holt verschiedene Parameter für das gesamte Dokument.
- `pdf_set_value`, `pdf_get_value`
Setzt und liefert verschiedene numerische Werte für das Dokument.

- `pdf_get_image_height`, `pdf_get_image_width`
Die Funktionen liefern die Höhe und Breite eines Bildes.
- `pdf_set_horiz_scaling`
Setzt die horizontale Skalierung bei der Textausgabe.
- `pdf_set_text_pos`
Setzt die Position, wo der nächste nicht positionierte Text erscheinen soll.
- `pdf_skew`
Verschiebt das Koordinatensystem.
- `pdf_continue_text`
Schreibt den Text in die nächste Zeile.
- `pdf_stringwidth`
Liefert die benötigte Breite einer Zeichenkette mit dem aktuellen Zeichensatz.
- `pdf_save`
Sichert die aktuelle Umgebung
- `pdf_restore`
Stellt eine zuvor gesicherte Umgebung wieder her
- `pdf_translate`
Setzt den Ursprung des Koordinatensystems innerhalb der Seite.
- `pdf_scale`
Setzt den Skalierungsfaktor.
- `pdf_rotate`
Setzt den Rotationswinkel für den Text.
- `pdf_setflat`
Setzt den maximalen Abstand in Pixeln zwischen einer Ausrichtungslinie und einer daran entlanglaufenden Konstruktion.
- `pdf_setlinejoin`
Setzt die Verbindungsart von Linien (eckig, rund usw.).
- `pdf_setlinecap`
Setzt den Type der Linienenden.

- `pdf_setmiterlimit`
Die Funktion bestimmt die Grenze der äußeren Zusammenführungslinie zweier Linien.
- `pdf_setlinewidth`
Setzt die Linienbreite.
- `pdf_setdash`
Setzt das Muster für gestrichelte Linien.
- `pdf_moveto`
Diese Funktion setzt den Ausgabezeiger auf eine bestimmte Position.
- `pdf_curveto`
Zeichnet eine Kurve bzw. ein Kreissegment.
- `pdf_lineto`
Zeichnet eine Linie vom aktuellen Punkt zu einem relativen Ziel.
- `pdf_circle`
Zeichnet einen Kreis.
- `pdf_arc`
Zeichnet einen Kreisbogen.
- `pdf_rect`
Zeichnet ein Rechteck.
- `pdf_closepath`
Schließt einen Zeichenpfad und vollendet damit die Darstellung.
- `pdf_stroke`
Zeichnet eine Linie entlang eines Pfades.
- `pdf_closepath_stroke`
Zeichnet eine Linie entlang des Pfades und schließt diese dann ab.
- `pdf_fill`
Füllt den aktuellen Pfad mit einer Farbe ohne Muster.
- `pdf_fill_stroke`
Zeichnet eine Linie entlang des Pfades und füllt den gesamten Pfad dann mit einer Farbe ohne Muster.

- `pdf_closepath_fill_stroke`
Schließt, füllt und zeichnet eine Linie entlang des Pfades.
- `pdf_endpath`
Beendet den aktuellen Pfad und vollendet damit die Darstellung.
- `pdf_clip`
Begrenzt alle Zeichenoperation auf den aktuellen Pfad.
- `pdf_setgray_fill`
Setzt die Füllfarbe auf einen Grauwert (0 bis 1).
- `pdf_setgray_stroke`
Setzt die Zeichenfarbe auf einen Grauwert (0 bis 1).
- `pdf_setgray`
Setzt die Zeichen- und Füllfarbe auf einen Grauwert (0 bis 1).
- `pdf_setrgbcolor_fill`
Setzt die Füllfarbe auf einen RGB-Farbwert. Dabei werden die Farbwerte als einzelne Parameter zwischen 0 und 1 übergeben.
- `pdf_setrgbcolor_stroke`
Setzt die Zeichenfarbe auf einen RGB-Farbwert. Dabei werden die Farbwerte als einzelne Parameter zwischen 0 und 1 übergeben.
- `pdf_setrgbcolor`
Setzt die Zeichen- und Füllfarbe auf einen Farbwert. Dabei werden die Farbwerte als einzelne Parameter zwischen 0 und 1 übergeben.
- `pdf_add_outline`
Fügt ein Lesezeichen zur aktuellen Seite hinzu.
- `pdf_set_transition`
Setzt den Übergang zur nächsten Seite für bewegte Bilder. Diese Funktion hat keinen Effekt beim Druck.
- `pdf_set_duration`
Setzt die Zeitdauer die vergeht, bis zur nächsten Seite geblättert wird. Diese Funktion hat keinen Effekt beim Druck.
- `pdf_open_image_file`
Liest ein Bild in den Formaten GIF, JPG, PNG oder TIFF aus einer Datei und bindet es ein. Die Darstellung erfolgt mit `pdf_place_image`.

- `pdf_open_memory_image`
Liest ein Bild, das mit PHP erzeugt wurde, aus dem Speicher.
- `pdf_close_image`
Schließt ein Bild.
- `pdf_place_image`
Platziert ein Bild auf der Seite.
- `pdf_put_image`
Speichert ein Bild im PDF-Dokument für die erneute Verwendung. Dies kann dann mit `pdf_execute_image` angezeigt werden.
- `pdf_execute_image`
Plaziert ein gespeichertes Bild auf der Seite.
- `pdf_add_annotation`
Fügt eine Anmerkung oder Notiz hinzu.
- `pdf_set_border_style`
Setzt den Stil der Umrandung von Verweisen und Anmerkungen
- `pdf_set_border_color`
Die Funktion setzt die Farbe der Umrandung von Verweisen und Anmerkungen
- `pdf_set_border_dash`
Setzt das Muster für gestrichelte Linien als Umrandung von Verweisen und Anmerkungen

9.3.3 PDF-Funktionen praktisch verwenden

Die PDF-Funktionen sind nicht besonders kritisch in der Anwendung, abgesehen von einige Schwierigkeiten, die Anfänger mit dem Koordinatensystem haben. Die folgenden Beispiele zeigen, wie einige interessante Funktionen verwendet werden.

Testumgebung für PDF-Funktionen

Um PDF zu sehen, müssen Sie den Acrobat Reader oder Acrobat ab Version 4.0 verwenden. Das Erzeugen von PDF-Dateien und Ablegen auf Festplatte ist relativ umständlich. Wenn Acrobat korrekt installiert ist, sollten auch die Plug-Ins für den Browser vorhanden sein, sodass die Anzeige direkt erfolgen kann. Der folgende Aufruf dient zum Starten des Skripts, dass die PDF-Dateien im Speicher erzeugt und direkt an den Browser sendet:

```
<a href="$PHP_SELF?script=pdf_generate&show=yes">
  Klicken Sie hier, um die PDF-Datei anzuzeigen
</a>
```

Der Parameter *\$pdfexample* definiert ein Skript, dass nur den Inhalt einer bereits vorbereiteten Seite erzeugt. Die »Rohseite« erzeugt folgendes Skript, dass in dieser oder ähnlicher Form immer verwendet werden kann:

```
<?php
class pdf {
    function pt2cm($px) {
        return round($px / 72 * 2.54, 2);
    }
    function cm2pt($cm) {
        return round($cm * 72 / 2.54, 0);
    }
}
$pdf = pdf_new();
pdf_open_file($pdf);
$width = pdf::cm2pt(21.0); # A4
$height = pdf::cm2pt(29.4); # A4
pdf_begin_page($pdf, $width, $height);

/* An dieser Stelle wird der Code eingebaut, der Elemente auf der
   PDF-Seite erzeugt. */

pdf_end_page($pdf);
pdf_close($pdf);
header("Content-type: application/pdf");
header("Content-Description: PHP Generated Data" );
echo pdf_get_buffer($pdf);
exit;
?>
```

Listing 9.9:
Standardskript zum
Erzeugen einer leeren
PDF-Seite
(*pdf_generate*)

Die Klasse *pdf* dient nur zur Definition zweier Umrechnungsfunktionen von Zentimeter in Punkt und umgekehrt. Das eigentliche Gerüst der PDF-Datei mit zwei Funktionen erzeugt:

Wie es funktioniert

```
$pdf = pdf_new();
pdf_open_file($pdf);
```

Danach steht die Datei im Speicher bereit, der Zugriff kann über das Handle in *\$pdf* erfolgen.

Die Seite wird dann unter Angabe der Größe in Punkt erzeugt:

```
pdf_begin_page($pdf, $width, $height);
```

Sind alle Elemente erzeugt worden, wird die Seite beendet und die Datei geschlossen:

```
pdf_end_page($pdf);
pdf_close($pdf);
```

Zuletzt werden die Header erzeugt, damit der Browser den Dateityp erkennen und Acrobat Reader starten kann.

```
header("Content-type: application/pdf");  
header("Content-Description: PHP Generated Data" );
```

Das Auslesen der im Speicher erzeugten Daten erfolgt mit der Funktion `pdf_get_buffer`:

```
echo pdf_get_buffer($pdf);
```

Die folgenden Beispiele stehen jeweils in der mit `include` eingebundenen Datei.

Text ausgeben

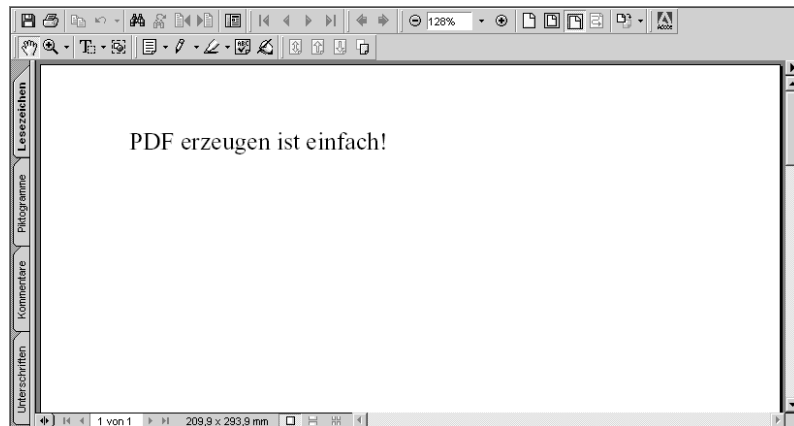
Das Ausgeben von Text

*Listing 9.10:
pdf_show:
Text ausgeben*

```
<?php  
pdf_set_font($pdf, "Times-Roman", 20, "host");  
$left = pdf::cm2pt(2.50);  
$up = pdf::cm2pt(29.5 - 2.50);  
pdf_set_text_pos($pdf, $left, $up);  
pdf_show($pdf, "PDF erzeugen ist einfach!");  
?>
```

Der Vorgang umfasst mehrere Schritte. Zuerst muss ein Font festgelegt werden, dann die Textposition. Die Ausgabe des eigentlichen Textes basiert dann auf diesen Vorgaben. Das Koordinatensystem einer PDF-Seite beginnt übrigens links unten und orientiert sich an der Grundlinie des Fonts, Unterlängen werden also abgeschnitten.

*Abbildung 9.8:
Text ausgeben*



Erzeugen eines Fließtextes in einer Box

Das nächste Beispiel zeigt, wie Fließtext über mehrere Zeilen erzeugt wird. Achten Sie auch hier wieder auf das Koordinatensystem. Der

Startpunkt des Textes ist wiederum links unten, was ohne vorherige Berechnung der Höhe des Textblocks schwer festzulegen ist.

```
<?php
pdf_set_font($pdf, "Times-Roman", 14, "host");
$left = pdf::cm2pt(2.50);
$up = pdf::cm2pt(29.5 - 5.50);
$width = pdf::cm2pt(5.0);
$height = pdf::cm2pt(5.0);
$mode = "justify";
$text = <<<PDF
Dieser Text wird deshalb hier so lang geschrieben, damit die
Funktion auch wirklich demonstriert werden kann.
PDF;
pdf_show_boxed($pdf, $text, $left, $up, $width, $height, $mode);
?>
```

Listing 9.11:
pdf_show_boxed:
Text in einer Box

In der folgenden Abbildung ist der Koordinatenursprung des Textes mit Pfeilen markiert, dieser Punkt muss ausgehend vom linken unteren Seitenrand berechnet werden, wenn der globale Koordinatenursprung nicht verschoben wurde.

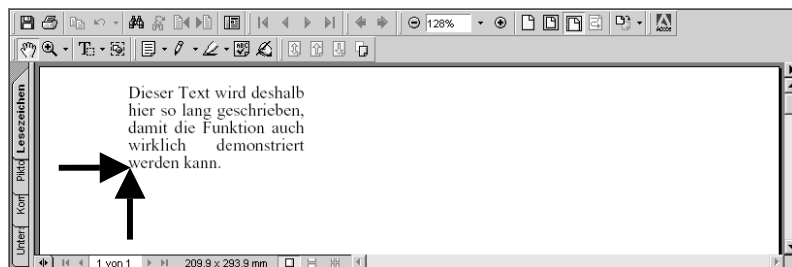


Abbildung 9.9:
Textbox erzeugen

Steuerung von Textauszeichnungen

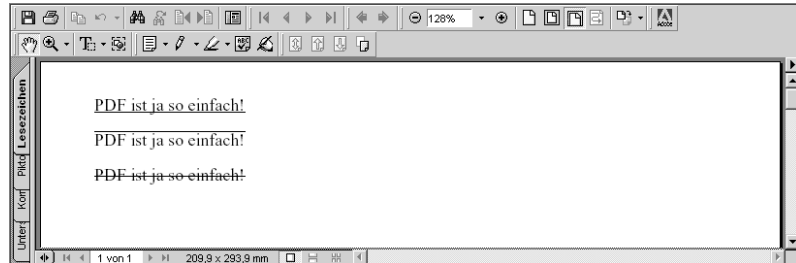
Verschiedene Textdekorationen lassen sich mit der Parameterfunktion `pdf_set_parameter` erzeugen. Das folgende Beispiel zeigt dies.

```
<?php
$tmp = array("underline", "overline", "strikeout");
pdf_set_font($pdf, "Times-Roman", 14, "host");
$left = pdf::cm2pt(1.50);
$up = pdf::cm2pt(29.5 - 1.50);
$delta = pdf::cm2pt(1);
$text = "PDF ist ja so einfach!";
foreach($tmp as $tmc) {
    pdf_set_parameter($pdf, $tmc, "true");      # setzen
    pdf_show_xy($pdf, $text, $left, $up);
    pdf_set_parameter($pdf, $tmc, "false");     # zurücksetzen
    $up -= $delta;
}
?>
```

Listing 9.12:
pdf_set_parameter:
Linie direkt erzeugen

Der Einsatz eines Arrays dient hier nur zur Vereinfachung der Programmierung. Beim Experimentieren können Sie den zweiten Parameter auch direkt setzen.

Abbildung 9.10:
Textdekorationen

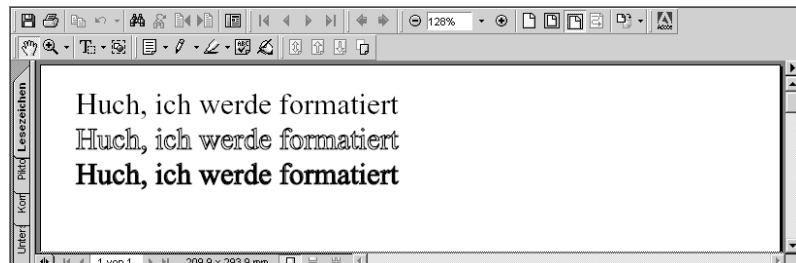


Das nächste Beispiel zeigt die Bearbeitung von Rändern von Buchstaben.

Listing 9.13:
pdf_set_rendering:
Textränder
formatieren

```
<?php
pdf_set_font($pdf, "Times-Roman", 24, "host");
$left = pdf::cm2pt(1.0);
$up = pdf::cm2pt(29.5 - 1.50);
$delta = pdf::cm2pt(1);
$text = "Huch, ich werde formatiert";
for ($i = 0; $i <= 3; $i++) {
    pdf_set_text_rendering($pdf, $i);
    pdf_show_xy($pdf, $text, $left, $up);
    $up -= $delta;
}
?>
```

Abbildung 9.11:
Textformatierung



Um fortlaufend Text zu erzeugen, eignet sich die Funktion `pdf_continue_text`. Die Anwendung wird im nächsten Beispiel gezeigt:

Listing 9.14:
pdf_continue_text:
Mehrzeiligen Text
ausgeben

```
<?php
function pdfout($message) {
    global $pdf;
    pdf_continue_text($pdf, $message);
}

$point = 14;
pdf_set_font($pdf, "Times-Roman", $point, "host");
$left = pdf::cm2pt(1.0);
$up = pdf::cm2pt(29.5 - 0.50);
pdf_set_leading($pdf, $point * 1.5);
```

```
pdf_set_text_pos($pdf, $left, $up);
$text = "Hier werden viele Buchstaben und Wörter verwendet.";
pdfout($text);
$text = "Nun geht es einfach weiter im Text";
pdfout($text);
$text = "Nun haben wir sogar einen Zeilenumbruch ohne Box.";
pdfout($text);
?>
```

Das »Weiterschalten« der Zeilen erfolgt hier automatisch, der Text muss nun nur noch ausgegeben werden. Beachten Sie, dass der Text von oben nach unten läuft, der Ursprung aber auch hier wieder unten liegt, und zwar am unteren Rand (der Grundlinie) der ersten Zeile.

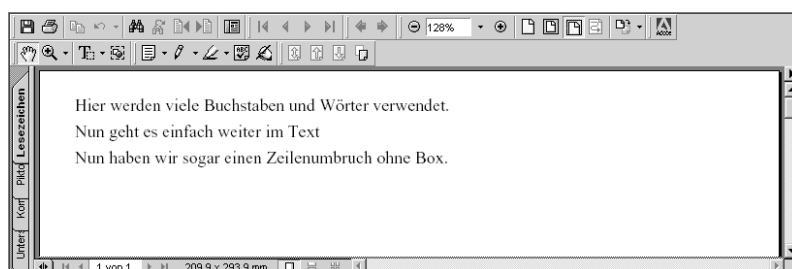


Abbildung 9.12:
Mehrere Zeilen Text
ausgeben

Linien zeichnen

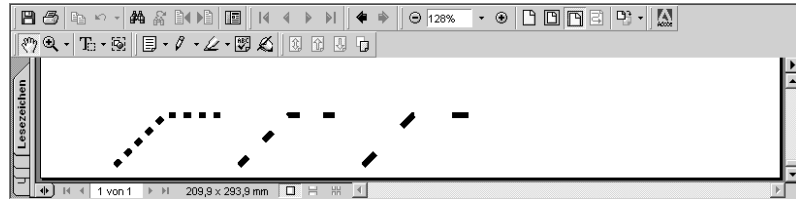
Für Linien und Figuren stehen einige Standardfunktionen zur Verfügung, wie sie auch einfache Zeichenprogramme liefern. Komplexe Formen verlangen freilich einigen Aufwand.

```
<?php
pdf_setlinewidth($pdf, 5);
for ($i = 1; $i <= 6; $i+=2) {
    pdf_setdash($pdf, 2 * $i + 3, 6 * $i + 1);
    pdf_moveto($pdf, 10 + $i * 50, 10);
    pdf_lineto($pdf, 50 + $i * 50, 50);
    pdf_lineto($pdf, 100 + $i * 50, 50);
    pdf_stroke($pdf);
}
?>
```

Listing 9.15:
pdf_lines: Linien
zeichnen

Beim Zeichnen von Linien über mehrere Punkte hinweg müssen Sie daran denken, die Form mit `pdf_stroke` abzuschließen. Erst danach erscheint sie auf der Seite. In PDF sind alle Figuren Objekte, die in sich geschlossen sind.

Abbildung 9.13:
Linie zeichnen



Weitere Beispiele

Viele weitere Beispiele finden Sie in der ausführlichen Referenz – »PHP 4. Die Referenz« – erschienen im Carl Hanser Verlag.

9.4 Ausgabesteuerung

Bei der Programmierung größerer Skripte kann es sinnvoll sein, die Ausgabe von Daten zum Browser zu kontrollieren. Mit PHP 4 ist dies nun komfortabel möglich.

9.4.1 Grundlagen der Pufferung

Normalerweise werden Daten, die PHP mit `echo` oder `print` erzeugt, ebenso wie reiner HTML-Code sofort an den Browser gesendet. Sie können deshalb auch nicht über diese Ausgaben verfügen. In PHP 3 gab es nur die Möglichkeit, global die so genannte Pufferung zu aktivieren. Dabei werden die Ausgaben nicht mehr sofort gesendet, sondern in einem internen Speicher gesammelt. Am Ende des Skripts oder zwischendurch mit der Funktion `flush` wird dieser Puffer geleert – die Ausgaben erscheinen im Browser.



In PHP 4 sind neue Funktionen verfügbar, mit denen Sie den Ausgabepuffer direkt kontrollieren können. Prinzipiell kann per Funktion die Pufferung aktiviert werden. Anschließend können Sie den Inhalt bearbeiten. Dazu sind folgende Eigenschaften verwendbar:

- Zugriff auf den Inhalt des Puffers
- Löschen des Puffers
- Starten und Beenden der Pufferung

9.4.2 Umgang mit den Pufferfunktionen

Die Funktionen selbst sind wenig spektakulär. Erst im Zusammenhang mit anderen Funktionen ergeben sich interessante Anwendungsmöglichkeiten. Sie können beispielsweise den generierten Inhalt in einer Datei speichern. Damit besteht eine Möglichkeit, die Aus-

gaben von Skripten später auf CD zu brennen und offline zur Verfügung zu stellen.

Funktionsübersicht

Die folgende Übersicht zeigt alle Pufferfunktionen; »ob« steht für »output buffer«:

- `ob_end_clean()`
Löscht den Puffer und beendet die Pufferung.
- `ob_end_flush()`
Beendet die Pufferung und sendet den gesamten Inhalt an den Browser.
- `ob_get_contents()`
Holt den Inhalt des Puffers und gibt ihn als Zeichenkette zurück.
- `ob_implicit_flush()`
Deaktiviert die Pufferung.
- `ob_start()`
Aktiviert die Pufferung.

`ob_end_clean()`
`ob_end_flush()`
`ob_get_contents()`
`ob_implicit_flush()`
`ob_start()`

Wurde der Puffer geleert, kann er natürlich mit `print`, `echo` oder anderen Ausgabefunktionen erneut beschrieben werden.

Anwendungsbeispiel

Das folgende Beispiel erzeugt eine kleine HTML-Datei und schreibt sie in ein Verzeichnis. Anschließend wird sie ausgegeben. Beachten Sie, dass das Skript mit Schreibrechten abgearbeitet werden muss.

```
<?php
ob_end_flush();
ob_start();
echo "Diese Datei (sample_buffer.txt) wurde dynamisch erzeugt\n";
$hdlFile = fopen("sample_buffer.txt", "w");
fwrite($hdlFile, ob_get_contents());
fclose($hdlFile);
flush();
?>
```

Listing 9.16:
Erzeugen einer
HTML-Datei mit
Pufferfunktionen:
buffer_all

Wenn Sie den Befehl in der letzten Zeile (`flush()`) wie folgt ersetzen, wird der Text nicht parallel zum Browser gesendet:

```
ob_end_clean();
```

Das folgende Beispiel zeigt die Wirkung des Puffers, indem mehrere Ausgaben zeitlich verzögert erscheinen, was nur ohne Puffer sichtbar wird:

*Listing 9.17:
Fortschrittsanzeige
mit ausgeschaltetem
Puffer: buffer_demo*

```
<?php
function delay() {
    $start = time();
    while ($start == time()) { }
}
ob_implicit_flush();
for ($i = 0; $i < 10; $i++, print "#", delay()) { }
?>
```

Das Skript in Listing 9.17 gibt die Symbole nebeneinander und im Abstand von etwa einer Sekunde aus; am Ende sehen Sie das folgende Bild:

```
#####
```

*Listing 9.18:
Wiederholung von
Inhalten:
buffer_demo2*

```
<?php
function delay() {
    echo ob_get_contents();
    echo "<br>";
    ob_end_flush();
    ob_start();
}
ob_start();
for ($i = 0; $i < 10; $i++, print "#", delay()) { }
?>
```

Dieses Skript gibt untereinander Pärchen von #-Zeichen aus. Dabei wird das erste Zeichen in den Puffer geschrieben (mit print), dann wird innerhalb der Verzögerungsfunktion der Inhalt des Puffers noch einmal ausgegeben (mit echo ob_get_contents). Anschließend wird der Zeilenumbruch erzeugt und der Puffer gesendet und gelöscht.

Weitere Einsatzmöglichkeiten

Im Projekt *phpTemple* werden die Pufferfunktionen verwendet, um die fertige HTML-Seite nachträglich zu modifizieren. Auch der Skriptgenerator, der auf der die als XML-Dateien kodierten Skripte in diesem Buch ausgibt, verwendet den Inhalt des Ausgabepuffers, um die Daten nachträglich so aufzubereiten, dass sie den Anforderungen entsprechen.

9.5 Reguläre Ausdrücke

Dieser Abschnitt bietet eine umfassende Darstellung der Anwendung von regulären Ausdrücken. Wenn Sie noch nicht mit regulären Ausdrücken vertraut sind, lesen Sie zuerst den ➡ Abschnitt 3.2.12

drücken vertraut sind, lesen Sie zuerst den ➡ Abschnitt 3.2.12 *Reguläre Ausdrücke* ab Seite 217. Die folgenden Darstellungen setzen voraus, dass Sie mit dem Begriff regulärer Ausdruck etwas anfangen und zumindest einfache Muster auch lesen können.

9.5.1 Einführung

Perl hat sich auch wegen der hervorragenden Unterstützung regulärer Ausdrücke durchgesetzt. Es war naheliegend bei der Entwicklung von PHP, eine solche Unterstützung ebenfalls zu bieten. Um PHP erfolgreich gegen Perl etablieren zu können, mussten auch die »Profis« entschädigt werden; PHP unterstützt fast alle Funktionen regulärer Ausdrücke wie in Perl. Statt einer festen Integration in den Sprachkern erfolgt die Verarbeitung jedoch mit speziellen Funktionen. Mit diesen nachfolgend beschriebenen Perl-kompatiblen Funktionen zur Behandlung regulärer Ausdrücke ist ein hervorragendes Konzept entstanden – einfache Anwendung und hohe Leistungsfähigkeit in einem.

➡ Abschnitt 9.5.6 *Ersetzungen* ab Seite 694 behandelt detailliert die Nutzung regulärer Ausdrücke für Ersetzungen mit der Funktion `preg_replace`.



9.5.2 Unterschiede zu Perl

Dieser Abschnitt wendet sich an Leser, die bereits Perl-Profis sind und mit regulären Ausdrücken umgehen können. Es werden lediglich die Unterschiede zu Perl 5.005 erläutert. Im Folgenden ist von der PCRE-Syntax bzw. einfach nur PCRE die Rede. PCRE steht für Perl Compatible Regular Expressions und damit sind immer die Funktionen in PHP gemeint, die diese Syntax unterstützen. Es gibt auch andere Funktionen regulärer Ausdrücke, auf die nicht mehr eingegangen wird, da sie veraltet und weniger leistungsfähig sind und in künftigen Versionen von PHP vermutlich nicht mehr unterstützt werden.

Profi-Tipps für Perl-Profis

Verhalten der Whitespace-Zeichen

Standardmäßig gilt als Whitespace ein Zeichen, wenn es von der C-Funktion `isspace()` erkannt wird. Normalerweise sind dies die Zeichen für *newline* (Neue Zeile), *space* (Leerzeichen), *formfeed* (Seitenvorschub), *carriage return* (Wagenrücklauf), *horizontal tab* (Horizontaler Tabulator) und *vertical tab* (Vertikaler Tabulator). Perl 5 unterstützt inzwischen den vertikalen Tabulator nicht mehr (dieser hat nur für Drucker eine Bedeutung). Da PHP als Quellcode vorliegt und selbst kompiliert werden kann, weicht die verwendete Zeichentabelle möglicherweise von dieser Liste und den in Perl vorherrschenden Bedingungen ab.



Testmuster

Die Testmuster verhalten sich nicht völlig Perl-konform. So würde man erwarten, dass der Ausdruck `(?!a){3}` bedeutet, dass die nächsten drei Zeichen nicht »a« sein dürfen. Tatsächlich nimmt PHP hier an, dass als Nächstes nicht drei Mal »a« auftreten darf, also nicht »aaa«.

Null-Zeichen (binäre 0) sind nicht erlaubt, da das Suchmuster als C-Zeichenkette gespeichert wird, die bekanntlich mit 0 abgeschlossen wird. Solche Zeichen müssen mit `\0` gekennzeichnet werden.

Nicht unterstützte Escape-Zeichen und andere Abweichungen

Escape-Zeichen

Die folgenden Escape-Zeichen werden nicht unterstützt:

- `\l`, `\u`, `\L`, `\U`, `\E`, `\Q`

Tatsächlich sind diese auch nicht Bestandteil der regulären Ausdrücke in Perl, sondern werden vom Perl-Zeichenkettenmodul ausgewertet. `\G` wird nicht unterstützt, da es für Einzelersetzungen nicht relevant ist. PCRE unterstützt keine Konstruktionen mit `{code}`.

Bekannte Unterschiede

Bei einigen Konstruktionen verhält sich Perl nicht, wie theoretisch zu erwarten wäre. Wenn Sie »aba« mit dem Muster `/^(a(b)?)+$/` testen, wird `$2` auf den Wert »b« gesetzt. Der Test von »aabbaa« mit dem Muster `/^(aa(bb)?)+$/` schlägt jedoch fehl. Dagegen funktioniert er mit dem Muster `/^(aa(b(b)))+$/`. Bei Perl 5.004 gingen noch beide Fälle, wie auch in PCRE. Sollten sich künftige Perl-Versionen wieder im Verhalten ändern, wird PCRE dem vermutlich folgen.

Beim Test von »a« gegen das Muster `/^(a)?(?(1)a|b)+$/` bringt Perl eine Übereinstimmung, PCRE dagegen nicht. Keine Übereinstimmung finden jedoch sowohl PCRE als auch Perl beim Test von »a« gegen das Muster `/^(a)?a/`.

Erweiterungen in PCRE

Obwohl der Vorwärtstest nur Zeichenketten konstanter Länge akzeptiert, können bei Rückwärtstests unterschiedliche Längen angewandt werden. Perl erwartet, dass auch diese Zeichenketten gleich lang sind.

Wenn `PCRE_DOLLAR_ENDONLY` gesetzt ist und `PCRE_MULTILINE` nicht, reagiert das `$`-Zeichen nur auf das absolute Ende der Zeichenkette (und nicht auf das nächste Zeilenende).

Wenn `PCRE_EXTRA` gesetzt ist, führt ein Backslash mit folgenden Buchstaben ohne spezielle Bedeutung zu einer Fehlermeldung.

Wenn `PCRE_UNGREEDY` gesetzt ist, wird die Gierigkeit des Wiederholungsoperators invertiert. Wenn jedoch ein `?`-Zeichen folgt, erfolgt dies nicht.

9.5.3 Übersicht

Folgende Funktionen sind in PHP für reguläre Ausdrücke verfügbar:

- `preg_match`
Führt eine normale Suche nach regulären Ausdrücken durch. Die Funktion stoppt, wenn der Ausdruck erstmalig erfüllt wurde.
- `preg_match_all`
Führt eine globale Suche nach regulären Ausdrücken durch. Es werden alle Übereinstimmungen gefunden und in einem Array zurückgegeben.
- `preg_replace`
Sucht und ersetzt reguläre Ausdrücke.
- `preg_replace_callback`
Sucht und ersetzt reguläre Ausdrücke unter Verwendung einer benutzerdefinierten Funktion.
- `preg_split`
Teilt eine Zeichenkette nach einem regulären Ausdruck und übergibt die Teile in ein Array.
- `preg_quote`
Versieht einige Sonderzeichen mit dem Auszeichnungssymbol, damit sie ihre spezielle Wirkung verlieren.
- `preg_grep`
Durchsucht ein Array anhand eines Musters und gibt die Elemente zurück, für die eine Übereinstimmung gefunden wurde.

Der Präfix *preg* steht für PERL REGULAR EXPRESSIONS. Wenn Sie sich tiefer mit regulären Ausdrücken beschäftigen möchten, lohnt auch ein Blick in gute Perl-Bücher.



Hinweis

9.5.4 Die Suchmuster im Detail

Ein regulärer Ausdruck ist ein Suchmuster, das auf eine untersuchte Zeichenkette von links nach rechts angewendet wird. Die meisten Zeichen in einem regulären Ausdruck repräsentieren lediglich sich selbst. Einige Sonderzeichen kennzeichnen Positionen und Wiederholungen.

Buchtipp:
Jeffrey E.F. Friedl,
»Reguläre
Ausdrücke«
O'Reilly (ISBN
3-930673-62-2)

Metazeichen

Diese einfachen Einsatzbeispiele wurden bereits in Kapitel 3 (➔ Abschnitt 3.2.12 *Reguläre Ausdrücke* ab Seite 217) gezeigt und sollen hier erneut detailliert behandelt werden. Sie sollten sich über die Bedeutung der folgenden Metazeichen im Klaren sein:

- \

Das »Escape«-Zeichen, leitet Sonderzeichen ein oder hebt die Sonderzeichenwirkung eines Zeichens wieder auf.

- ^

Beginn einer Zeile oder des gesamten Musters.

- \$

Ende eines Wortes oder einer Zeile (im Mehrzeilenmodus).

- .

Steht für ein beliebiges Zeichen.

- [

Beginnt eine Zeichendefinition. Das Ende bildet die schließende eckige Klammer].

Einige Zeichen innerhalb der []-Klammern, die eine Zeichenklasse bilden, haben eine besondere Bedeutung:

- \

Definiert ein Zeichen als Sonderzeichen (normalerweise ist die Bedeutung der Sonderzeichen aufgehoben).

- ^

Negiert die Auswahl.

- -

Wird als Bereichsoperator eingesetzt (beispielsweise [a-z]).

-]

Beendet die Zeichenklassendefinition.

- |

Startet einen alternativen Zweig (ODER-Bedingung).

- ()

Eine untergeordnete Suchmustergruppe oder eine einfache Gruppierung. Klammern müssen immer paarweise auftreten. Für die Zählung der Gruppen ist die öffnende Klammer entscheidend.

- `?`
Steht für kein oder ein Zeichen.
- `*`
Steht für kein oder beliebig viele Zeichen.
- `+`
Steht für ein oder mehr Zeichen.
- `{n,m}`
Bezeichnet eine minimale Anzahl `n` oder maximale Anzahl `m` Zeichen.

Die Musterschalter der regulären Ausdrücke

Die folgende Liste erläutert alle PCRE-Optionen. Diese Optionen wirken auf den gesamten Ausdruck, modifizieren also das Verhalten insgesamt. Sie werden außerhalb des Suchmusters hinter das schließende Begrenzungszeichen geschrieben. Die Symbole selbst unterscheiden Groß- und Kleinschreibung.

PCRE-Optionen

- `i` (PCRE_CASELESS)
Ignoriert Groß- und Kleinschreibung im Suchmuster.
- `m` (PCRE_MULTILINE)
Normalerweise wird der Suchausdruck als eine Zeichenkette betrachtet, auch wenn es mehrere Zeilen sind. Das Startzeichen `^` wirkt nur auf den absoluten Anfang, `$` nur auf das absolute Ende. Wird diese Option gesetzt, gilt jeder Zeilenanfang und jedes Zeilenende als Start und Ende für Suchmuster. Auf Zeichenketten, die keine `»\n«` enthalten, hat dies keinen Einfluss.
- `s` (PCRE_DOTALL)
Wenn diese Option gesetzt ist, erfüllt der Punkt (`.`) alle Zeichen, auch `»\n«`.
- `x` (PCRE_EXTENDED)
Wird diese Option gesetzt, werden alle Whitespaces²⁷ ignoriert.
- `e`
Mit dieser Option werden die Angaben in der Ersatzzeichenkette bei `preg_replace` als PHP-Code interpretiert.

²⁷ Whitespaces sind: Leerzeichen, Tabulatoren, Zeilenumbrüche.

- A (PCRE_ANCHORED)
Schränkt die Suche auf den Anfang der Zeichenkette ein.
- D (PCRE_DOLLAR_ENDONLY)
Schränkt die Suche auf das Ende der Zeichenkette ein. Wird ignoriert, wenn gleichzeitig `m` genutzt wird.
- S
Wird ein Muster mehrfach verwendet, kann die Ausführungsgeschwindigkeit gesteigert werden, indem diese Option eingesetzt wird. PHP speichert den übersetzten Ausdruck dann zwischen.
- U (PCRE_UNGREEDY)
Bestimmt die Arbeitsweise der Musteranweisungen bezüglich der Mehrfachauswahl. Normalerweise sind reguläre Ausdrücke gierig, finden also soviel wie möglich – U dreht dieses Verhalten um und der Ausdruck findet nur soviel wie nötig.

Die Bedeutung des Backslash

\ (Backslash)

Das Zeichen `\` hat eine besondere Stellung. Es definiert folgende Zeichen in ihrer Bedeutung um. Wenn es von einem nicht alphanumerischen Zeichen gefolgt wird, wird die besondere Bedeutung dieses Zeichens unterdrückt. Dies gilt auch innerhalb einer Zeichenklassendefinition. Der Backslash selbst wird mit `\\` dargestellt.

Unsichtbare Zeichen

Eine andere Anwendung ist die Darstellung nicht sichtbarer Zeichen, beispielsweise des Tabulators: `\t`. Außerdem können Zeichen in kodierter Form angegeben werden:

- `\a`
Das Alarm- oder BEL-Zeichen (hex 07).
- `\b`
Der Backspace bzw. Rückschritt (hex 08).
- `\cx`
»Control-x«, x kann jedes Standardzeichen sein. Die Umwandlung des Zeichens geschieht folgendermaßen: Kleinbuchstaben werden in Großbuchstaben gewandelt. Dann wird das Bit 6 (hex 40) invertiert. Beispiele:
 - `\cz` wird zu 1A
 - `\c{` wird zu 3B
 - `\c;` wird zu 7B

- `\e`
Das Escape-Zeichen (hex 1B).
- `\f`
Steht für Formfeed (das Seitenumbruchzeichen, hex 0C).
- `\n`
Das Newline (das Zeilenumbruchzeichen hex 0A).
- `\r`
Steht für Carriage return oder Wagenrücklauf (hex 0D).
- `\t`
Ersatzzeichen für Tab oder Tabulator (hex 09).
- `\xhh`
Zeichen mit Hexcode *hh*. Groß- und Kleinschreibung der Hexadezimalziffern A–F spielt keine Rolle. Nur die nächsten beiden Zeichen nach `\x` werden ausgewertet.
- `\0ddd`
Dies definiert Zeichen im Oktalcode *ddd*. Werden weniger als drei Ziffern angegeben, werden auch nur diese gelesen. Um eindeutige Resultate zu erzielen, sollte mit führenden Nullen aufgefüllt werden: `\011`. Generell ist es nicht zwingend, die führende Null zu setzen, praktisch ist es aber dringend zu empfehlen.

Folgt dem Backslash eine Ziffer zwischen 1 und 9, wird diese Folge als Referenz erkannt. Referenzen werden später beschrieben. Folgen größere Zahlen, werden diese als Oktalzahlen erkannt und daraus wird ein Byte generiert. Beispiele:

- `\040` ergibt ein Leerzeichen (eine oktale 40 ist eine hexadezimale 20 oder eine dezimale 32 oder einfach ein ASCII-Leerzeichen).
- `\40` ist mit `\040` identisch, wenn weniger als 40 Referenzen existieren. (Achtung: Dies ist schwer zu verstehen, aber die Referenzen werden dezimal gezählt, die Interpretation der Zahl dagegen okt.)
- `\7` ist immer eine Referenz (größer als 0 und kleiner als 10).
- `\11` ist ein Tabulator und entspricht `\t` (ASCII-Code 9).
- `\0113` ist ein Tabulator, der von einer »3« gefolgt wird. Hier ist die Zuordnung eindeutig, da nur 99 Referenzen zulässig sind.
- `\377` ist ein Byte, bei dem alle Bits 1 sind (hex FF).

Probleme mit Referenzen und Zahlen



Beispiel

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

- \81 kann die 8. Referenz sein, wenn eine solche existiert oder eine binäre 0, gefolgt von den Ziffern »8« und »1«.

Alle Zeichenfolgen, die ein einzelnes Byte ergeben, können innerhalb und außerhalb von Zeichenklassendefinitionen genutzt werden. Allerdings haben die Zeichen mit dem Escape-Symbol \ eine besondere Bedeutung, die bereits gezeigt wurde.

Generische Zeichenklassen

Eine weitere Bedeutung des Backslash besteht in der Darstellung generischer Zeichenklassen:

- \d

Erkennt eine Dezimalziffer.

- \D

Erkennt alles, was keine Dezimalziffer ist.

- \s

Jedes Whitespace (nicht sichtbares Zeichen).

- \S

Jedes Zeichen, das kein Whitespace ist.

- \w

Jedes »Wortzeichen«, also Buchstaben, Ziffern und der Unterstrich. Als Buchstabe gilt die Auswahl entsprechend der lokalisierten Zeichentabelle. Die Buchstaben »äöü« usw. werden dann als solche erkannt, wenn die lokale Zeichentabelle diese auch enthält.

- \W

Jedes Zeichen, das kein Buchstabe, Ziffer und Unterstrich ist.

Eine weitere Bedeutung des Backslashes besteht in der Definition bestimmter Aussagen. Damit werden für das betroffene Muster Bedingungen genannt; das Zeichen selbst nimmt jedoch keine Zeichenposition ein. Die möglichen Bedeutungen werden nachfolgend beschrieben:

- \b

Eine Wortgrenze. Diese wird dadurch erkannt, dass das aktuelle und das vorhergehende Zeichen nicht zugleich \w oder \W entsprechen, also \w auf \W folgt oder umgekehrt.

- \B

Keine Wortgrenze.

- \A

Beginn eines Musters, unabhängig vom Mehrzeilenmodus. Dies unterscheidet sich vom Zeichen `^` insofern, als nur der absolute Beginn akzeptiert wird, egal was andere Bedingungen anzeigen.

- \Z

Ende eines Musters oder Ende einer Zeile, unabhängig vom Mehrzeilenmodus. Dies unterscheidet sich vom Zeichen `$` insofern, als nur das absolute Ende akzeptiert wird, egal was andere Bedingungen anzeigen.

- \z

Ende eines Musters.

Diese Aussagen gelten nicht innerhalb von Zeichenklassendefinitionen, die teilweise dieselben Symbole enthalten, ihnen dort aber eine andere Bedeutung verleihen.

Die Symbole Circumflex (^) und Dollar (\$)

Das `^`-Zeichen wurde bereits betrachtet. Es definiert entweder den Anfang eines Musters oder, innerhalb einer Zeichenklassendefinition, eine Negation der Klasse. Im Gegensatz zu den klassischen Beispielen muss das Zeichen aber nicht am Anfang stehen, wenn mehrere Alternativen mit `|` definiert werden und der Anfang nur für eine dieser Alternativen erkannt werden soll. Dies führt mitunter zu schwer lesbaren Ausdrücken. Beginnen aber tatsächlich alle Alternativen mit dem `^`-Zeichen, spricht man von einem verankerten Muster (engl. *anchored pattern*).

Das `$`-Zeichen steht dagegen für das Ende eines Musters. Es hat sonst keine weitere Bedeutung. Mit der Bedingung `D` (`PCRE_DOLLAR_ENDONLY`) kann festgelegt werden, ob nur das absolute Ende der Zeichenkette erkannt wird oder das Ende untergeordneter Suchmuster. Auf `\Z` hat dies keine Auswirkungen.

Die Bedeutung der Zeichen `^` und `$` kann sich ändern, wenn im Mehrzeilenmodus gearbeitet wird. Normalerweise geht man bei regulären Ausdrücken davon aus, dass eine Zeichenkette bearbeitet wird. Manche Texte, wie beispielsweise HTML-Seiten, bestehen aber aus mehreren Zeilen. Auch solche Gebilde können mit regulären Ausdrücken in einem Zug bearbeitet werden.

Mehrzeilenmodus

Beachten Sie, dass komplexe Ausdrücke, die auf große Textmengen angesetzt werden, erhebliche Systemleistung und einiges an Ausführungszeit benötigen. Reguläre Ausdrücke sind fast alles, nur nicht schnell.



Wenn die Option `PCRE_MULTILINE` gesetzt ist, ändert sich die Bedeutung der Zeichen `^` und `$`. In diesem Fall wird mit `^` als Anfang der absolute Beginn der Zeichenkette oder das erste Zeichen einer neuen Zeile – nach dem Newline-Zeichen `\n` – erkannt. Ebenso wird mit `$` als Ende das absolute Ende der Zeichenkette erkannt, aber auch das letzte Zeichen vor dem Zeilenumbruch mit `\n`. Beispiel:



- Das Muster `/^abc$/` erfüllt die Zeichenkette `"def\nabc"`, wenn es im Mehrzeilenmodus abgearbeitet wird, nicht aber im Standardmodus.

`PCRE_DOLLAR_ENDONLY` wird ignoriert, wenn `PCRE_MULTILINE` gesetzt ist. Die Symbole `\A`, `\Z` und `\z` funktionieren dagegen in beiden Modi.

Punkt

. (Punkt)

Außerhalb einer Zeichenklassendefinition steht der Punkt für ein Zeichen, egal ob Buchstabe, Ziffer, Sonderzeichen, sichtbar oder nicht. Wird die Option `PCRE_DOTALL` gesetzt, wird auch der Zeilenumbruch `\n` als Zeichen angesehen.

Innerhalb einer Zeichenklassendefinition steht der Punkt für sich selbst, darf also nicht zusätzlich von einem Backslash angeführt werden.

Zeichenklassendefinitionen

[]

Die eckigen Klammern `[` und `]` enthalten eine Zeichenklassendefinition. Darunter wird eine spezifische Zeichenfolge verstanden. Eine solche Definition kann für eines oder mehrere Zeichen des Suchmusters stehen. Aufeinanderfolgende Zeichen können mit einem Minuszeichen gebildet werden. Beispiele:



- `[0-9a-fA-F]`
Dieser Ausdruck definiert Hexadezimalziffern.
- `[0-7]`
Definiert Oktalziffern.
- `[01]`
Definiert Binärziffern.
- `[02468]`
Definiert gerade Ziffern.
- `[aeiouAEIOU]`
Definiert Vokale, jedoch ohne Umlaute.

Wenn die Klammer selbst Bestandteil der Klasse sein soll, sollte sie mit dem Backslash kombiniert werden:

- `[\\[]]`

Definiert die eckigen Klammern als Klasse.

Als erstes Zeichen kann das `^`-Zeichen gesetzt werden, dann wird die Definition negiert:

- `[^aeiouAEIOU0-9]`

Definiert Konsonanten durch Ausschluss von Vokalen und Ziffern. Zusätzlich sollten noch die Sonderzeichen ausgeschlossen werden – dieser Ausdruck ist also noch nicht ganz perfekt.

Diese Funktion hat das `^`-Zeichen nur, wenn es das erste Zeichen ist, ansonsten steht es für sich selbst. Um die Wirkung sicher zu unterdrücken, stellen Sie ein Backslash `\` voran.

Der Zeilenumbruch wird als normales Zeichen behandelt. Eine leere Zeile mit einem Zeilenumbruch am Ende wird mit `^[a]` erkannt. Die Optionen `PCRE_DOTALL` und `PCRE_MULTILINE` spielen keine Rolle.

Die Anwendung des Minuszeichens als Bereichsoperator wurde bereits gezeigt. Wenn Sie diese Zeichen in die Klassendefinition einbeziehen möchten, müssen Sie ein Backslash davor setzen. Das erste und das letzte Zeichen der Bereichsangabe sind jeweils inklusive. Im Zusammenhang mit den eckigen Klammern ergeben sich einige Fallen. So kann die schließende Klammer `]` nicht als Ende eines Bereiches angegeben werden, dem weitere Definitionen in der Klasse folgen. Betrachten Sie die folgenden Beispiele:

- `[X-]23]`

Definiert eine Klasse mit den Zeichen »X« und »-«, gefolgt von den alleinstehenden Zeichen »2«, »3« und »]«. Erfüllende Zeichenketten wären »X23]« oder »-23]«.

- `[W-\]46]`

Definiert den Bereich von »W« bis »]«, dies ist zugleich die gesamte Klasse, gefolgt von den Zeichen »46«.

Bereiche in Zeichenklassendefinitionen richten sich nach der Folge der Zeichen im ASCII-Zeichensatz. Es können auch numerische Folgen angegeben werden, beispielsweise mit Oktalzahlen:

- `[\000-\037]`

Dieser Ausdruck definiert die Zeichen 0 bis 31 (die dezimale 31 entspricht der oktalen 37).



Problematisch werden Klassen, wenn die Bereichsgrenzen Groß- und Kleinbuchstaben enthalten, die Beachtung von Groß- und Kleinschreibung aber unterdrückt wird:



- `[W-c]` entspricht `[] [\^_`wxyzabc]`
- `[\xc8-\xcb]`

Dies entspricht allen E-Zeichen mit Akzent im Französischen, also Ê, É usw.

Die Zeichen `\d`, `\D`, `\s`, `\S`, `\w` und `\W` dürfen auch innerhalb einer Zeichenklassendefinition verwendet werden:

- `[\dABCDEF]`

Entspricht Hexadezimalziffern mit Großbuchstaben.

- `[\^W_]`

Entspricht allen Buchstaben und Ziffern, aber nicht dem Unterstrich.

Die nicht alphanumerischen Zeichen `\`, `-` und `^` innerhalb der Definition haben an manchen Stellen keine besondere Bedeutung. Dann ist die Verwendung des Backslash optional.

Senkrechter Strich

Mit dem senkrechten Strich werden Alternativen getrennt. Beispiel:

- `Kennwort|password`

Dieses Suchmuster erfüllt sowohl die Zeichenkette »Kennwort« als auch »password«. Eine Klammerung ist nicht notwendig, solange die Schreibweise eindeutig ist.

Es gibt keine Beschränkung in der Anzahl aufeinanderfolgender Alternativen. Es ist auch erlaubt, eine leere Zeichenkette anzugeben. Der Suchalgorithmus arbeitet das Muster von links nach rechts ab. Wird eine Übereinstimmung gefunden, bricht er ab. Weitere Elemente, die eventuell Zeichenbehandlungsroutinen enthalten, werden dann nicht mehr ausgeführt.

Interne Schalter

Die Schalter haben Sie bereits im ➡ Abschnitt *Die Musterschalter der regulären Ausdrücke* auf Seite 673 kennengelernt. Es ist möglich, diese Schalter auch innerhalb der Suchmuster zu setzen, also mit begrenzter Gültigkeit. Dazu wird die spezielle Sequenz `(?o)` verwendet, wobei `o` für eine der folgenden Schalter steht:

- `i` steht für `PCRE_CASELESS`
Keine Beachtung von Groß- und Kleinschreibung
- `m` steht für `PCRE_MULTILINE`
Mehrzeilenmodus ist aktiviert.
- `s` steht für `PCRE_DOTALL`
Der Punkt erfüllt auch Zeilenumbruchzeichen.
- `x` steht für `PCRE_EXTENDED`
Erweiterte Darstellung mit ignorierten Leerzeichen.
- `U` steht für `PCRE_UNGREEDY`
Hebt das gierige Verhalten des Ausdrucks auf.

Die anderen Schalter können innerhalb des Ausdrucks nicht eingesetzt werden, weil sie immer global wirken.

Das Beispiel verdeutlicht den Einsatz:

- `(?im)`
Steht für Mehrzeilenmodus ohne Beachtung von Groß- und Kleinschreibung.



Es ist auch möglich, Optionen wieder zu negieren, diesmal mit dem Minuszeichen:

- `(?im-sx)`
Dies setzt die Optionen `PCRE_CASELESS` und `PCRE_MULTILINE` und löscht die Optionen `PCRE_DOTALL` und `PCRE_EXTENDED`.

Erscheint ein Optionsbuchstabe zugleich vor und hinter dem Minuszeichen, wird die Option gelöscht. Die Reichweite der Wirkung hängt davon ab, wo die Sequenz steht. Innerhalb eines Teilmusters spielt die Position keine Rolle, die folgenden Muster sind identisch:

- `((?i)abc)`
- `(a(?i)bc)`
- `(ab(?i)c)`
- `(abc(?i))`

Solche Angaben gelten also für das gesamte Suchmuster. Wenn innerhalb des Musters mehrere sich widersprechende Optionen auftreten, gewinnt die auf der rechten Position stehende.

In Teilmustern gilt, dass die Option nur auf das Teilmuster wirkt. Dies ist in PHP anders als in Perl 5 definiert.

- `(a(?i)b)c`

Dieses Muster findet »aBc« und »abc«, aber keine anderen Kombinationen aus Groß- und Kleinbuchstaben.

- `(a(?i)b|c)`

Hier werden die Zeichenketten »ab«, »aB«, »c« und »C« erkannt. Dies erscheint nicht logisch, liegt aber an der Art, wie der Interpreter den Ausdruck übersetzt. Er wird zuerst die Optionen erkennen – (?i) zwingt ihn, Großbuchstaben zu akzeptieren –, dann wird er den Ausdruck abarbeiten; (?i) wirkt also auch auf den alternativen Zweig.

Die Optionen `PCRE_UNGREEDY (?U)` und `PCRE_EXTRA (?X)` können ebenso gesetzt werden. (?X) sollte am besten am Anfang stehen, da sich die Verhaltensweise folgender Optionen danach richtet.

Teilmuster

()

Innerhalb des Suchmusters können Teilmuster gebildet werden. Diese werden durch runde Klammern (und) angegeben. Zwei Dinge werden damit erreicht:

- Ein Satz Alternativen wird zusammengefasst.
- Teilmuster werden als einzelne Referenzen zurückgegeben. Ausschlaggebend für die Referenznummer ist die öffnende Klammer, die von links beginnend gezählt wird. Dies ist bei tieferen Verschachtelungen wichtig.

Einige Beispiele:



- `Katze(njammer|nstreu)`

Dieses Muster findet die Worte »Katze«, »Katzenjammer« oder »Katzenstreu«, nicht jedoch »Katzen«.

- `Katze(njammer|nstreu|)`

Dieses Muster entspricht dem vorhergehenden, findet aber außerdem eine leere Zeichenkette.

- `the ((white|black)(king|queen))`

Ergibt drei Teilmuster, wenn die Zeichenkette »The black king« geprüft wird: »black king«, »black« und »king«. Diese Teilmuster haben die Referenzen 1, 2 und 3.

Die Tatsache, dass runde Klammern mehrere Aufgaben übernehmen, führt beim Lesen regulärer Ausdrücke zu großen Problemen. Zudem sind oft mehrere Lösungen möglich. Sie können die Referenzierung mit der Angabe `?:` unterdrücken, um die Stellen zu dokumentieren,

die nicht gezählt werden müssen. Außerdem lassen sich so Teilmuster gezielt aus der Referenzierung ausblenden. Ein Beispiel:

- `the ((?:white|black)(king|queen))`

Dieses Muster ergibt nur zwei Teilmuster, wenn die Zeichenkette »The white queen« geprüft wird: »white queen«, »queen«. Diese Teilmuster haben die Referenzen 1 und 2. Das Teilmuster »white« wird unterdrückt.

Die maximale Anzahl von referenzierten Teilmustern beträgt 99. Die maximale Anzahl von Teilmustern an sich dagegen beträgt 200. In die Angabe der Unterdrückung der Referenz kann auch die Optionswahl eingebaut werden. Die folgenden beiden Suchmuster sind identisch:

- `(?:i:montag|mittwoch)`
- `(?:(?:i)montag|mittwoch)`

Beide Muster finden »montag« und »MONTAG« und jede andere Kombination von Klein- und Großbuchstaben. In beiden Fällen wird die Referenzierung unterdrückt – es stehen keine Teilmuster zur Verfügung.

Beachten Sie auch hier, dass eine Option in einem Zweig oder Teilmuster auf die anderen Teile Auswirkungen hat, da die Optionen zuerst ausgewertet werden und danach erst die Abarbeitung der Muster von links nach rechts erfolgt.

Wiederholungen

Alle Optionen, die Sie bisher kennen gelernt haben, können wiederholt werden:

- Einzelne Zeichen, mit oder ohne Backslash
- Der Punkt
- Zeichenklassendefinitionen
- Referenzen
- Geklammerte Teilmuster (wenn es keine Bedingung ist)

Referenzen und Bedingungen werden später noch beschrieben.

Generell werden Wiederholungsanweisungen in geschweifte Klammern gesetzt. Der höchstmögliche Wert ist 65 536. Die obere und untere Grenze zulässiger Wiederholungen wird mit einem Komma getrennt:

- `z{2,4}`

Dieses Muster findet »zz«, »zzz« und »zzzz«.



V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

- `[aeiou]{3,}`

Hiermit finden Sie Anordnungen mit mindestens 3 Vokalen.

- `\d{8}`

Steht für genau acht Ziffern.

Beachten Sie bei der Angabe von Sonderzeichen, dass eine alleinste-hende, schließende geschweifte Klammer *kein* Sonderzeichen ist, son- dern nur sich selbst repräsentiert. Ebenso wird durch eine alleinste- hende, öffnende geschweifte Klammer kein Fehler hervorgerufen, wenn in diesem Kontext Wiederholungen nicht erlaubt sind. Die Klammer steht auch dann nur für sich selbst.

- `{,6}`

Dieses Muster ist keine Wiederholungsanweisung, sondern steht für die vier Zeichen »{«, »«, »6« und »}«.

- `{0}`

Dies ist dagegen erlaubt und steht für kein Zeichen, was im all- gemeinen aber sinnlos ist.

Drei Abkürzungen des universellen Wiederholungsoperators kennen Sie bereits:

- `*` entspricht `{0,}`
- `+` entspricht `{1,}`
- `?` entspricht `{0,1}`



Warnung

Sie können Endlosschleifen produzieren (tun Sie das nicht!), wenn Sie eine Zeichenkette, die nichts enthält, unendlich wiederholen: `(a?)*`. Es gibt seltene Fälle, in denen dies dennoch sinnvoll ist, deshalb wird der Interpreter das nicht als Fehler melden.

Greedy und Nongreedy



Insider

Das englische Wort »greedy« steht für »gierig«. Damit sind Muster gemeint, die so viele Übereinstimmungen wie möglich zurückgeben. Ein klassisches Beispiel ist die Erkennung von Kommentarzeichen in C. Kommentare in C stehen zwischen `/*` und `*/`. Ein naheliegendes Suchmuster wäre:

- `*.**`

Die Anwendung auf die folgende Zeile

```
/* Kommentar 1 */ Kein Kommentar /* Kommentar 2 */
```

funktioniert aber nicht, denn der Bestandteil `.*` nimmt die gesamte Zeichenkette als erfüllt an; er verhält sich »gierig«. Der Punkt steht für

jedes Zeichen und dies schließt diejenigen Zeichen, die den schließenden Teil der Klammer erkennen sollen, mit ein. Das Muster ist damit »übererfüllt«.

Richtig wäre dieses Muster:

- `*.*?*`

Hier wird das Verhalten mit dem folgenden Fragezeichen modifiziert. Damit wird nicht die Bedeutung modifiziert, sondern die Zählung der Übereinstimmungen. Dies ist eine andere Anwendung, als Sie bisher kannten. Ein weiteres Beispiel verdeutlicht dies:

- `\d??\d`

Dieses Muster erkennt eine Übereinstimmung, wenn eine Ziffer gefunden wurde. Es erkennt diese auch, wenn zwei Ziffern gefunden wurden, aber nur dann, wenn der Rest des Musters nur erfüllt werden kann, wenn zwei Ziffern gefunden wurden.

Mit der Option `PCRE_UNGREEDY` kann das Verhalten *non greedy* erzwungen werden, auch wenn kein `?`-Operator angegeben wird. Diese Option kennt Perl nicht.

Wenn ein geklammertes Teilmuster mit einer minimalen Anzahl Wiederholungen ausgestattet ist, die größer als 1 oder mit einem Maximalwert belegt ist, wird mehr Speicherplatz benötigt im Verhältnis zum angegebenen Minimum oder Maximum. Startet ein Muster mit `.*` oder `{0,}` und die Option `PCRE_DOTALL` ist gesetzt (der Punkt stimmt auch mit `\n` überein), wird der Ausdruck implizit verankert. Es spielt keine Rolle, was noch folgt, der Ausdruck beginnt immer links mit einem Zeichen. Eine andere Möglichkeit, dieses Verhalten zu erreichen, ist die Voranstellung eines `\A`.

Wenn ein referenziertes Teilmuster wiederholt wird, wird die letzte Übereinstimmung ausgewertet:

- `(diedel[die]{3}\s*)+`

Dieses Muster stimmt unter anderem mit »dideldie diedeldei« überein, als Teilmuster wird aber nur »dideldei« zurückgegeben.

- `(a|(b))+`

Dieses Muster stimmt unter anderem mit »aba« überein, als zweite Referenz wird »b« zurückgegeben.

Rückwärts-Referenzen (Back Reference)

Ich hoffe, dass Sie den bisherigen Ausführungen problemlos folgen konnten. Am besten spielen Sie etwas mit verschiedenen Ausdrücken,



um sicher im Umgang damit zu werden. Die folgenden Darstellungen stellen etwas höhere Ansprüche.

Außerhalb einer Zeichenklassendefinition kann auf referenzierte Teilmuster (diese wurden im letzten Abschnitt behandelt) verwiesen werden. Das Muster verweist damit partiell auf Teile seiner selbst. Diese Rückreferenz wird mit der Zeichenfolge `\N` erreicht, wobei *N* für eine Ziffer zwischen 1 und 9 steht. Die Zählung erfolgt anhand der Anzahl der öffnenden (linken) runden Klammern der Teilmusterausdrücke.

Ein Fehler wird nur dann erzeugt, wenn die Anzahl der Klammern geringer ist als die durch eine Rückreferenz verlangte Ziffer. Stimmt die Anzahl, wird kein Fehler erzeugt, wenn die Referenz nichts enthält. Dabei kann die Referenz an jeder Stelle stehen, die Teilmuster müssen sich nicht links davon befinden. Auf die Auswahl mit Hilfe des Backslash wurde weiter oben bereits eingegangen. Dieser Abschnitt zeigt den Umgang mit den Referenzen an sich.

Eine Rückreferenz findet Übereinstimmungen mit einem Suchmuster entsprechend dem referenzierten Teilmuster. Ein Beispiel:



- `(Kapital|Kommun) und \1ismus`

Dieses Muster findet »Kapital und Kapitalismus« und »Kommunismus«, aber nicht »Kapitel und Kommunismus«.

- `((?i)bla)\s+\1`

Hier wird »bla bla« und »BLA BLA« gefunden, aber nicht »BLA bla«.

Referenzen können vom Suchwort abhängen. Dies ist ein leichter Programmierfehler, der nur schwer zu erkennen ist. Das folgende Beispiel funktioniert – manchmal:

- `(a|(bc))\2`

Wenn als Suchmuster »bc« steht, funktioniert die Referenzierung mit `\2`, bei »a« dagegen nicht.

Es sind generell 99 rückwärtige Referenzen erlaubt. Alle Ziffern hinter einem Backslash werden deshalb als Referenznummer gewertet. Um unmittelbar folgende Ziffern abzugrenzen, muss ein Leerzeichen oder Kommentar gesetzt werden. Leerzeichen sind nur erlaubt, wenn die Option `PCRE_EXTENDED` gesetzt wurde.

Eine Referenz kann nicht in einem Teilmuster eingesetzt werden, das zu einer Definition führt, `(a\1)` beispielsweise funktioniert deshalb nicht. Wiederholungen sind jedoch erlaubt:

- $(a|b\backslash 1)^+$

Dieses Muster bringt Übereinstimmung mit jeder beliebigen Anzahl von »a« und »aba«, »ababaa« usw.

Beachten Sie, dass Referenzen innerhalb von Zeichenklassendefinitionen nicht erlaubt sind, ein manchmal schwerwiegender Verlust, der zwar die Universalität von Ausdrücken reduziert, dafür aber die Lesbarkeit steigert. Letzteres ist aber sicher nicht der Maßstab, denn gut lesbar sind reguläre Ausdrücke sowieso nie.

Bedingungen

Eine Bedingung wird für das vorhergehende oder folgende Zeichen ($?$ = eines zu untersuchenden Zeichens aufgestellt. Sinngemäß entspricht ($?!$ dies Formulierungen wie »Stimme mit x überein, wenn vor/nach x etwas bestimmtes steht«. Einige besonders einfache Bedingungsoperatoren wurden bereits beschrieben:

- $\backslash b, \backslash B, \backslash A, \backslash Z, \backslash z, \wedge, \$$

Hier werden komplexere Bedingungen vorgestellt. Grundsätzlich wird zwischen Bedingungen unterschieden, die nach der aktuellen Position (*look ahead*, vorausschauend) oder vor der aktuellen Position (*look behind*, zurückschauend) testen.

Die Funktionsweise entspricht allen anderen Abfragekonstruktionen auch, nur die Position des Abfragezeigers im Suchwort wird nicht beeinflusst. Bedingungen werden mit dem $?$ -Zeichen eingeleitet:

- $(?=$

Leitet eine vorausschauende, übereinstimmende Bedingung ein. Das nächste Zeichen muss also übereinstimmen, damit die Bedingung wahr wird. Das Zeichen selbst wird jedoch in der weiteren Analyse nicht mehr betrachtet.

- $(?!$

Leitet eine vorausschauende, nicht übereinstimmende Bedingung ein. Das nächste Zeichen darf nicht übereinstimmen, damit die Bedingung wahr wird. Das Zeichen selbst wird jedoch in der weiteren Analyse nicht mehr betrachtet.

Einige Beispiele zeigen, wie dies angewendet wird:

- $\backslash w+(?=;)$

Dieses Muster sucht Wörter, die von einem Semikolon gefolgt werden, bezieht das Semikolon selbst jedoch nicht in die Auswertung mit ein.



- `foo(?!bar)`

Hier werden Übereinstimmungen mit dem Wort »foo« gesucht, das nicht von dem Wort »bar« gefolgt werden darf.

- `(?!foo)bar`

Dieses Muster findet dagegen eine Übereinstimmung nicht, die mit etwas anderem als »foo« beginnt. Es findet aber jede Übereinstimmung mit »bar«, weil die Sequenz `(?!foo)` immer wahr ist.

`(?<=`
`(?<!`

Das Dilemma des letzten Musters ist nur mit weiteren Operatoren zu lösen. Die vorgestellten Beispiele bezogen sich auf vorausschauende Bedingungen. Folgende Symbole dienen der Definition zurückschauender Bedingungen:

- `(?<=`

Leitet eine zurückschauende, übereinstimmende Bedingung ein.

- `(?<!`

Leitet eine zurückschauende, nicht übereinstimmende Bedingung ein.

Das letzte Beispiel funktioniert besser folgendermaßen:



- `(?<!foo)bar`

Hier wird jede Übereinstimmung von »bar« gefunden, die nicht dem Wort »foo« folgt.

Bei dieser Form muss die Länge der abhängigen Zeichenkette mit der zu testenden identisch sein:



- `(?<=bullock|donkey)` ist falsch.
- `(?<!dogs?|cats?)` ist zulässig.

Ein weiteres Beispiel verdeutlicht dies und zeigt, wie man damit umgeht:



- `(?<=ab(c|de))`

Dieses Muster funktioniert nicht, da die alternativen Teilmuster »c« und »de« unterschiedlich lang sind.

- `(?<=abc|abde)`

Dies dagegen erfüllt den Zweck, denn nun wurde nur ein Teilmuster definiert.

Die Formulierung der rückwärtigen Bedingungen könnte man so aufschreiben: »Nimm jede Alternative, gehe jeweils um die Länge des alternativen Suchmusters zurück und prüfe das Muster erneut«. Es ist

logisch, dass eine fehlende Übereinstimmung der Länge zu keiner Übereinstimmung der Muster führen kann.

- `(?<=\d{3})(?!999)foo`

Dieses Muster erkennt Suchworte, bei denen das Wort »foo« drei Ziffern folgt, die aber nicht »999« sein dürfen. Übereinstimmungen gibt es für »352foo« oder »111foo«.

- `(?<=(?!foo)bar)baz`

Dieses Muster erfüllt das Wort »barbaz«, aber nicht »foobarbaz«.

Wenn Sie Bedingungen bilden, stehen diese in runden Klammern. Dies sind keine Teilmuster und sie werden auch nicht referenziert. Sie dürfen auch nicht wiederholt werden; dies wäre sinnlos, denn entweder stimmt es überein oder nicht. Wiederholungen ändern nichts am Ergebnis. Stehen allerdings Teilmuster in Bedingungen, werden diese gezählt und referenziert, wenn es positive Bedingungen sind.

Einmalige Teilmuster

Wenn eine Unter- und eine Obergrenze für Wiederholungen angegeben wird, führt eine fehlende Übereinstimmung zu einem wiederholten Test, bis alle durch die Wiederholung gebildeten Muster geprüft wurden. Dies soll manchmal verhindert werden, um unnütze und zeitraubende Tests zu vermeiden, wenn das Ergebnis absehbar ist. Sehen Sie sich das folgende Beispiel an:

- Das Muster `\d+foo` wird auf die Zeichenkette »123456bar« angewendet.

Offensichtlich gibt es keine Übereinstimmung. Dennoch dauert die Untersuchung lange, denn der reguläre Ausdruck wird zuerst für alle sechs Ziffern untersucht, dann wird das Wort »bar« verglichen – mit negativem Ergebnis. Dann startet der Ausdruck erneut, diesmal mit nur fünf Ziffern usw. Insgesamt wird der Test sechsmal durchlaufen, obwohl schon beim ersten Mal erkannt werden sollte, dass eine Übereinstimmung nicht möglich sein wird. Die Ursache liegt in der konsequenten Abarbeitung von links nach rechts, mit der das Programm zur Auflösung regulärer Ausdrücke arbeitet.

Der Ausdruck funktioniert zwar, ist aber nicht elegant programmiert. (**?>**) Es gibt die Möglichkeit, das Verhalten mit dem folgenden Operator zu beeinflussen:

- `(?>`

Dieser Code befiehlt dem Interpreter, sofort aufzugeben, wenn keine Übereinstimmung gefunden wurde.





Beispiel

Ein weiteres Beispiel verdeutlicht dies:

- `(?>\d+)bar`

Eine andere Beschreibung dieses Typs ist die Annahme, dass diese Konstruktion einer alternativen Zeichenkette entspricht, die am aktuellen Auswertepunkt verankert ist, etwa so: `\d\d\d\d\d\bar`, wobei die Anzahl der Ziffern dennoch variabel ist, was eben ohne eine derartige Konstruktion nicht geht.

Solche einmaligen Teilmuster sind keine zählenden Teilmuster, werden also auch nicht referenziert. Dafür können die Konstruktionen verschachtelt und mit Bedingungen kombiniert werden.

Ein einfaches Muster wie `abcd$` wird auf eine lange Zeichenkette angewendet, die keine Übereinstimmung bietet. Hierbei sucht der Interpreter nach »a«, findet er eines, wird der Rest auf die folgenden Zeichen getestet. Bei dem Suchmuster `^.abcd$` wird hingegen sofort die gesamte Zeichenkette getestet. Wird keine Übereinstimmung gefunden, wiederholt sich der Vorgang mit einem Zeichen weniger usw. Besser wird es mit `^(?>.*)(?<=abcd)`. Hier wird nicht rückwärts die Suchfolge verringert. Die eingeschlossene Bedingung führt zum Abbruch der Suche, wenn die Zeichenfolge nicht sofort gefunden wird. Bei langen Suchworten kann dies zu einer deutlichen Steigerung der Suchleistung führen.

Bedingte Teilmuster

`(?(cond)...)`

Es ist möglich, Teilmuster insgesamt von einer Bedingung abhängig zu machen. Dabei wird das Teilmuster nur dann untersucht, wenn die Bedingung zutreffend ist. Dies lässt sich auf die Auswahl aus zwei Alternativen erweitern. Die Schreibweise ähnelt dem einfachen Bedingungsoperator in PHP:

- `(?(condition)ja-muster)`
- `(?(condition)ja-muster|nein-muster)`

Wenn die Bedingung erfüllt ist, wird das ja-muster ausgeführt, andernfalls das nein-muster. Werden mehr Alternativen angegeben, wird ein Laufzeitfehler erzeugt.

Es gibt zwei Arten von Bedingungen. Besteht die Zeichenfolge zwischen den Klammern aus einer Ziffernfolge, dann ist die Bedingung erfüllt, wenn das mit der jeweiligen Ziffer referenzierte Suchmuster erfüllt ist. Beispiel:



Beispiel

- `(\ ()? [^()]+ (?(1) \))`

Dieses Suchmuster ist schon eines der »gehobenen« Art. Um es zu analysieren, muss es in seine Bestandteile zerlegt werden. Der erste Teil enthält eine optionale öffnende Klammer `(\ ()?`. Wenn eine solche

Klammer existiert, wird diese zum ersten referenzierten Teilmuster. Dann folgen eines oder mehrere Zeichen, die keine Klammer sein dürfen: `[^()]+`. Der dritte Teil stellt die Bedingung dar: `(?(1)\)`. Die Bedingung prüft, ob das erste referenzierte Teilmuster übereinstimmt, in diesem Fall also, ob eine führende Klammer erkannt wurde. Wenn dies der Fall ist, und nur dann, wird auch auf eine schließende Klammer untersucht.

Als Bedingung ist neben Ziffern, die auf referenzierte Teilmuster verweisen, auch eine vorausschauende oder rückwärtige Bedingung (siehe oben) erlaubt. Ein weiteres Beispiel:

- `(?(?=[^a-z]*[a-z])\d{2}[a-z]{3}-\d{2}|\d{2}-\d{2}-\d{2})`

Diese Bedingung (fett) ist eine übereinstimmende vorausschauende Bedingung, der eine Reihe von Ziffern folgt, an die ein Buchstabe anschließen soll. Wird ein Buchstabe gefunden, wird er gegen die erste Alternative, sonst gegen die zweite getestet. Das Beispiel erkennt Suchwörter der Art »dd-aaa-dd« oder »dd-dd-dd«, wobei »d« einer Ziffer und »a« einem Buchstaben entspricht.



Kommentare

Kommentare erleichtern das Programmiererleben erheblich. Reguläre Ausdrücke sind kleine Programme und zur Dokumentation sind deshalb Kommentare sinnvoll. Zum einen lassen sich damit Ausdrücke besser beschreiben, zum anderen können Kommentare als offensichtliche Trennzeichen eingesetzt werden. Kommentare werden wie folgt geschrieben:

- `(?#Hier steht der Kommentar)`

Wenn die Option `PCRE_EXTENDED` gesetzt ist, kann außerhalb einer Zeichenklassendefinition ein Kommentar allein mit dem `#`-Zeichen eingeleitet werden. Der nächste Zeilenumbruch beendet den Kommentar dann. In diesem Fall werden Whitespaces ignoriert, die zur Auflockerung integriert wurden. Wenn Sie explizit Whitespaces suchen, müssen Sie diese nun natürlich in Zeichenklassen deklarieren.

Hinweise zur Leistung

Reguläre Ausdrücke sind nicht nur in der Programmierung komplex. Auch die Umsetzung und Auswertung stellt für den PHP-Interpreter eine Herausforderung dar. Die Ausdrücke werden zwar kompiliert und dann erheblich schneller ausgeführt, komplexe Konstrukte mit langen Zeichenketten können dennoch spürbar Zeit in Anspruch nehmen – auch auf schnellen Maschinen. Eine Webanwendung ist kaum praktikabel, wenn ein regulärer Ausdruck allein eine Sekunde zur Bearbeitung benötigt.



**Effiziente
Ausdrücke**

Einige Konstrukte benötigen aufgrund der internen Umsetzung erheblich mehr oder weniger Zeit als andere. Die folgende Auflistung zeigt die wesentlichsten Unterschiede. Da sich oft mehrere Wege zum Ziel finden lassen, besteht fast immer eine Möglichkeit, die Leistung eines Ausdrucks ohne Funktionseinbußen zu erhöhen.

- Es ist effektiver, eine Zeichenklassendefinition wie `[aeiuo]` zu nutzen, als eine Serie von Alternativen `(a|e|i|o|u)`.
- Wenn ein Ausdruck mit `.*` beginnt und die Option `PCRE_DOTALL` gesetzt ist, erkennt PHP, dass die Zeichenkette nur am Anfang stehen kann, und verankert sie implizit. Ist `PCRE_DOTALL` nicht gesetzt, kann diese Optimierung nicht ausgeführt werden.
- Wenn die Option `PCRE_DOTALL` nicht erwünscht ist, kann alternativ `^.*` gesetzt werden.
- Generell hilft es dem Interpreter, wenn die Beschreibung so direkt wie möglich erfolgt, implizite Annahmen werden oft erst erkannt, wenn der gesamte Ausdruck ausgewertet wird.
- Ausdrücke werden nicht weiter ausgewertet, wenn das Ergebnis feststeht. Bei der Abarbeitung sollte der wahrscheinlichste Fall also am weitesten links stehen.
- Wenn Sie mehrere Varianten eines Ausdrucks haben und diesen in einer häufig durchlaufenen Schleife anordnen müssen, testen Sie die Abarbeitungszeit der Varianten und wählen Sie den schnellsten.

9.5.5 Die PCRE-Funktionen im Detail

Mit den PCRE-Funktionen können Sie reguläre Ausdrücke in vollem Umfang nutzen. Zuvor ein genereller Hinweis zur Syntax: Die Ausdrücke erwarten – wie auch in Perl üblich – Begrenzungszeichen:

```
|^ausdruck$|
```

Dabei spielt es keine Rolle, ob Sie wie gezeigt senkrechte Striche oder Backslash oder irgendein anderes Zeichen nutzen. Es sollte nur im Weiteren nicht innerhalb des Ausdrucks vorkommen. Eingeführt wurden diese Begrenzungszeichen, um die Schalter am Ende (`i`, `e` usw.) abtrennen zu können.

Suchen mit Suchmustern**`preg_match()`**

Die Funktion `preg_match` durchsucht eine Zeichenkette nach einem regulären Ausdruck und gibt `TRUE` zurück, wenn das Muster gefunden wurde.

```
<?php
// Ermittelt die Seitenzahl in einem Satz
if (preg_match("/Seite\s+(\d+)/i","Gehe zu Seite #9.",$parts)){
    print "Die Seitenzahl ist $parts[1]";
} else {
    print "Seite nicht gefunden.";
}
?>
```

Listing 9.19:
Beispiel preg_match1

Mit der Funktion `preg_match_all` suchen Sie global nach einem Suchmuster. Wenn Gruppen gebildet wurden, werden die Ergebnisse in einem Array zurückgegeben, jede Gruppe erzeugt ein Element. Wurde ein Muster gefunden, wird die Suche fortgesetzt. Ein vierter Parameter kann die Zuordnung beeinflussen:

preg_match_all()

- `PREG_PATTERN_ORDER`

Die Funktion verhält sich wie `preg_match` (Standard). Die Reihenfolge entspricht den Fundstellen.

- `PREG_SET_ORDER`

Diese Option sortiert die Ergebnisse so, dass die Indizes mit den Gruppen übereinstimmen.

```
<?php
$string = "<b>Beispiel: </b><div align=left>Ein Test</div>";
echo "Zu untersuchende Zeichenkette:
    <p><code>".htmlspecialchars($string);
echo "</code><p>";
preg_match_all("<[>]+>(.*?)</>+>|U", $string, $out,
    PREG_PATTERN_ORDER);
echo '1: ' . htmlspecialchars($out[0][0])."<br>";
echo '2: ' . htmlspecialchars($out[0][1])."<br>";
echo '3: ' . htmlspecialchars($out[1][0])."<br>";
echo '4: ' . htmlspecialchars($out[1][1])."<br>";
?>
```

Listing 9.20:
Beispiel preg_match2

Sie können dieses Muster gut einsetzen, um Tags aus einer HTML-Seite zu extrahieren. Das Beispiel hat folgende Ausgabe:

```
Zu untersuchende Zeichenkette:
<b>Beispiel: </b><div align=left>Ein Test</div>
1: <b>Beispiel: </b>
2: <div align=left>Ein Test</div>
3: Beispiel:
4: Ein Test
```

Abbildung 9.14:
Zerlegen von HTML-
Tags

Im nächsten Beispiel werden typische amerikanische Telefonnummern gesucht.

Listing 9.21:
Suche von Dreier-
gruppen, die von
einer Vierergruppe
gefolgt werden:
preg_match_all

```
<?php
$numbers = "Call 555-1212 or 1-800-766-3456";
preg_match_all("/\d{3}\d{3}\d{3}\d{3}\d{3}/x",
               $numbers, $phones);
echo 'Folgende Nummern wurden erkannt:<br />';
foreach($phones[0] as $pn) echo "- $pn<br />";
?>
```

Die Ausgabe zeigt, dass der Ausdruck eine dreistellige Ziffernfolge am Anfang erwartet, die führende 1 (Landeskennzahl) wird deshalb abgetrennt:

Abbildung 9.15:
Die Nummern
wurden extrahiert

```
Folgende Nummern wurden erkannt:
- 555-1212
- 800-766-3456
```

9.5.6 Ersetzungen mit regulären Ausdrücken

preg_replace()

Alle vorangegangenen Beschreibungen regulärer Ausdrücke gingen davon aus, dass eine Zeichenkette anhand eines Musters – dem regulären Ausdruck – durchsucht werden soll. Im Ergebnis geben die entsprechenden Funktionen nur Wahr oder Falsch zurück – das Muster wurde erfüllt oder nicht.

Sehr viel häufiger wird jedoch die Ersetzung der gefundenen Teile einer Zeichenkette benötigt. Bei der Ersetzung nach regulären Ausdrücken werden erfüllte Suchmuster durch Ersatzzeichenketten ersetzt. Dies klingt verhältnismäßig einfach. Reguläre Ausdrücke wären nicht gefürchtet, wenn es nicht auch bei den Ersetzungen eine Reihe spezieller Funktionen gäbe.

So kann man auf die durch Klammern gebildeten Gruppen in der Ersetzung verweisen, indem jede Gruppe mit der Zeichenfolge `\n` adressiert wird, wobei `n` für die Nummer der öffnenden Klammer steht. Das folgende Beispiel zeigt dies:

Listing 9.22:
preg_replace1:
Ersetzungen mit
preg_replace

```
<?php
$string = "ASP PHP Perl JSP Ruby";
echo preg_replace( "/^(.*)\s(.*)\s(.*)\s(.*)\s(.*)$/U",
                  "\2 \3 \5<hr>", $string );
echo preg_replace( "/^(.*)\s(.*)\s(.*)\s(.*)\s(.*)$/U",
                  "\1 \4<hr>", $string );
echo preg_replace( "/^(.*)\s(.*)\s(.*)\s(.*)\s(.*)$/U",
                  "\0<hr>", $string );
?>
```

Dieses Beispiel gibt folgendes aus:

Abbildung 9.16:
Ersetzungen mit
Referenzen

```
PHP Perl Ruby
ASP JSP
ASP PHP Perl JSP Ruby
```

Der Zugriff auf die Position der Ersatzzeichenkette erfolgt mit den Operatoren `\1` bzw. `\2`. Welche Ziffer hier steht, bestimmt die Gruppe, wobei die öffnenden Klammern zählen. `\0` erlaubt den Zugriff auf den gesamten Ausdruck.

Wie es funktioniert

Die Funktion `preg_replace` sucht nach einem regulären Ausdruck und ersetzt die Fundstelle.

`preg_replace()`

Innerhalb der Ersatzzeichenkette kann mit `\n` auf die Gruppen im Muster Bezug genommen werden, wobei `n` eine Zahl zwischen 0 und 99 ist, `\0` bezieht sich auf die gesamte Zeichenkette. Der Beginn einer Gruppe wird mit einer runden Klammer eingeleitet.

Alle Parameter können auch Arrays sein. Die Suche wird dann in jedem Element des Arrays ausgeführt und ein Array wird auch zurückgegeben.

Die Option `/e` behandelt den Code in *replacement* als PHP-Code. Ist der Code fehlerhaft, wird ein Parser-Fehler ausgegeben. Die Option ist neu in PHP 4.



Das folgende Beispiel erkennt ein amerikanisches Datumsformat (»Jahr-Monat-Tag«) und wandelt es in einem Schritt in ein deutsches Format um:

```
<?php
$pattern = "/(19|20)(\d{2})-(\d{1,2})-(\d{1,2})/";
$replace = "\\4.\\3.\\1\\2";
$return = preg_replace($pattern, $replace, $date);
echo "Das Ergebnis der Ersetzung: $return";
?>
```

*Listing 9.23:
Beispiel `preg_replace`:
Ersetzen mit re-
gulären Ausdrücken
(Ausschnitt)*

Dieses Beispiel zeigt Folgendes aus, wenn `$date` »1999-5-27« enthält:

```
27.5.1999
```

Das nächste Skript macht den Inhalt aller HTML-Tags in der Zeichenkette `$html_body` zu Großbuchstaben, die Schleife gibt den Text dann aus:

```
<?php
$html_body = file('scripts/pcr_replaceeval.xml');
$html_tags = preg_replace("/(<\/?)(\w+)([>]*>)/e",
    "'\1'.strtoupper('\2').'\3'",
    $html_body);
foreach ($html_tags as $tag) {
    echo htmlspecialchars($tag)."<br>";
}
?>
```

*Listing 9.24:
`pcr_replaceeval`:
Ersetzen mit Aufruf
einer zusätzlichen
Funktion*

Abbildung 9.17:
Ersetzen mit
Funktionsaufruf

```
<SCRIPT>
<NAME>preg_replaceeval</NAME>
<TITLE>Ersetzungen in regulären Ausdrücken mit Funktionen</TITLE>
<DESCRIPTION>
Dieses Skript liest die eigene XML-Quelldatei und gibt diese formatiert aus.
Daf&amp;uuml;r m&amp;uuml;ssen die HTML-Tags
umgewandelt werden, was die Funktion <FUNCTION>htmlspecialchars</FUNCTION>
&amp;uuml;bernimmt. Mit der Funktion <FUNCTION>preg_replace</FUNCTION>
werden zuvor alle Tags in Gro&amp;szlig;buchstaben verwandelt (in der
Originaldatei werden nur Kleinbuchstaben verwendet).
</DESCRIPTION>
<CHAPTER>9</CHAPTER>
<LISTING>24</LISTING>
<OPTIONS standalone="yes"/>
<?php
$html_body = file('scripts/pcr_replaceeval.xml');
$html_tags = preg_replace("/(<\?)(\w+)(\[^\>]*\>)/e", "'\1'.strtoupper(
'\2').'\3'", $html_body);
foreach ($html_tags as $tag) {
echo htmlspecialchars($tag)."<BR>";
}
?>
</SCRIPT>
```

preg_split()

preg_split zerlegt eine Zeichenkette anhand eines regulären Ausdrucks. Zurückgegeben wird ein Array mit den Teilen, die maximale Anzahl kann mit einem dritten, optionalen Parameter eingeschränkt werden.

Listing 9.25:
Beispiel preg_split:
Zeichenketten
zerlegen

```
<?php
$words = "hypertext,programming ASP:PHP;Perl";
$keywords = preg_split("/[\\s,:;]+/", $words);
foreach ($keywords as $part) {
echo "$part<br />";
}
?>
```

preg_quote()

Die Funktion preg_quote markiert alle Sonderzeichen, die in regulären Ausdrücken eine Bedeutung haben, sodass sie ihre Bedeutung verlieren. Dazu wird ein Backslash davor gesetzt. Die betroffenen Zeichen sind:

```
. \ + * ? [ ^ ] $ ( ) { } = ! < > | :
```

preg_grep()

Die Funktion preg_grep gibt in einem Array die Elemente eines Arrays zurück, für die eine Übereinstimmung mit dem regulären Ausdruck gefunden wurde. Mit Hilfe der Arrayfunktionen kann dies leicht negiert werden:

Listing 9.26:
preg_grep:
Arrays analysieren

```
<?php
$array = array('Muster', 12, 'Haus', 35, 'Tag', 9);
$array = preg_grep("/^(\\d)+$/", $array);
$notinarray = array_diff($array, $notinarray);
echo "<u>Enthalten</u>:<br>";
foreach($notinarray as $element) echo "$element<br>";
echo '<br>';
echo "<u>Nicht enthalten</u>:<br>";
foreach($notinarray as $element) echo "$element<br>";
?>
```

Im Beispiel wird ein Array auf numerische Elemente hin untersucht. Die Abbildung zeigt das Ergebnis:

```
Enthalten:  
12  
35  
9  
  
Nicht enthalten:  
Huster  
Haus  
Tag
```

Abbildung 9.18:
Analyse eines Arrays
mit regulären
Ausdrücken

9.6 Extensible Markup Language – XML

XML ist mehr als ein Schlagwort. Tatsächlich verbirgt sich hinter diesem Begriff eine ganze Technologie. Komplexe Projekte und Geschäftsanwendungen profitieren erheblich davon. Auch für kleinere Projekte und den privaten Gebrauch finden sich interessante Anwendungsmöglichkeiten.

9.6.1 Was ist XML?

XML steht für »Extensible Markup Language« und beschreibt ein Datenformat für strukturierte Dokumente. Der Standard ist unter folgender Adresse definiert:

<http://www.w3.org/XML>

Die Festlegung als Standard und die schlichte Aussage, XML sei ein potenzieller Nachfolger für HTML, kann den überraschenden Erfolg von XML nicht recht erklären. Der ist auch umso erstaunlicher, als die praktische Anwendung bis vor kurzem überhaupt nicht, und auch heute noch nur sehr eingeschränkt, möglich ist. Es existiert nämlich kein Browser, der XML vollständig darstellen kann. Nur der Internet Explorer ab Version 5 kann wenigstens in Ansätzen damit umgehen und manchmal funktioniert auch das eine oder andere Beispiel. Dies mag verwunderlich erscheinen, wurde der Standard doch schon Anfang 1998 verabschiedet und seit dieser Zeit wird XML heftig diskutiert. Am Büchermarkt sind unzählige Publikationen zu finden und keine Fachzeitschrift, die nicht die neuesten Ergüsse der Autoren zu XML wiedergibt. Tatsache ist aber, dass die Funktion XML erst durch eine spezielle Anwendung mitgegeben wird – und die kann der Browser nicht kennen. Intelligenter sind serverseitige Lösungen, wie die hier im Buch vorgestellten Skripte auf PHP-Basis.

Dabei hat der Hype seinen Grund. Als wesentliches Merkmal wird die Möglichkeit hervorgehoben, dass mit XML eigene Tags definiert werden können. Das ist grundsätzlich richtig. Vergessen wird dabei (aus gutem Grund), was es bedeutet, eigene Tags zu erstellen. Das logische System eines richtigen, funktionalen und von Abhängigkeiten geprägten Tags ist außerordentlich komplex. Dennoch ist die Trennung von Daten in eine Daten- und eine Darstellungsebene mithilfe einer einfachen Auszeichnungssprache genial, denn dies öffnet völlig neue Per-



spektiven bei der computergestützten und damit automatisierten Verarbeitung von Daten. Erst wenn die Semantik eines Textes vom Computer erkannt werden kann, »versteht« dieser den Inhalt. XML definiert semantische Beschreibungselemente für dedizierte Anwendungsgebiete. Das ist die eigentliche Revolution.

So komplex freilich muss ein eigenes Projekt nicht sein. Aber es existieren heute schon viele Datenquellen in einem der vielfältigen XML-Formate. Die Verarbeitung derartiger Quellen kann auch mit PHP erfolgen.

XML und PHP

Die Nutzung von XML mit PHP ist eine neuartige Anwendung der Sprache. Denn hier wird erstmals eine Unabhängigkeit vom Browser erreicht. Sie können mit Hilfe von PHP einen eigenen XML-Parser schreiben und zur Ausgabe von einfachem HTML veranlassen. Dies ist in Anbetracht der derzeitigen Browser ein enormes Potenzial. Einfacher wird es deshalb nicht, denn Sie müssen dennoch XML erstellen können – einen Validator enthält PHP derzeit nicht.

Wie Sie in den XML-Zug einsteigen

Der Weg zu einer XML-Applikation ist nicht einfach. Allerdings wird die Schaffung einer Anwendung mit PHP erst möglich, wenn Sie einige Grundlagen verstanden haben. Einige Eigenschaften sind grundlegender Natur:

- XML definiert eigene Tags, diese können auch eigene Attribute besitzen.
- Im XML-Format kann man Daten speichern und bereitstellen.
- XML kann mit der Style-Sprache XSL (Extensible Stylesheet Language) formatiert werden.
- XML kann mit XSLT (XSL Transformation) in andere Formate (auch in andere XML-Dialekte) transformiert werden.
- XML trennt konsequent Daten und Darstellung im Browser.

Einige andere Eigenschaften, die XML nicht hat, sind ebenso wichtig:

- XML ist kein Ersatz zu HTML, sondern eine übergeordnete Definition für Auszeichnungssprachen. Es gibt mit XHTML eine Implementierung von HTML in XML.
- XML kann nicht zur Entwicklung darstellender Tags genutzt werden, dies ist die Aufgabe von CSS, XSL und XSL-FO (XSL Formatting Objects).

- XML geht strenger mit den Tags um und generiert bewusst Fehler.

XML verwendet bestimmte Dateierweiterungen, an denen der Browser erkennt, was für ihn bestimmt ist: **Dateierweiterungen**

- XML. XML-Datei; diese enthält die Sprachanweisungen.
- XSL. Style-Sheet für XML; hier wird die Ausgabe formatiert.
- CSS. Style-Sheet für HTML; auch hier wird die Ausgabe formatiert.
- DTD. Document-Type-Definition-Datei; hier werden die Tags definiert.
- JS. JavaScript-Datei; wird oft eingesetzt, um dynamische Funktionen zu erzeugen und Funktionssammlungen auszulagern.
- HTML. Kennen Sie sicher bereits!

Mit diesem Wissen und einem Editor ausgerüstet, sollten Sie nun Ihre erste XML-Datei erstellen:

```
<?xml version="1.0" ?>
<AUSGABE>Hallo XML!</AUSGABE>
```

Listing 9.27:
Einfache XML-Datei
[simple.xml](#)

Geben Sie den Text ein und schauen Sie sich das Ergebnis dann im Internet Explorer an.

```
<?xml version="1.0" ?>
<AUSGABE>Hallo XML!</AUSGABE>
```

Abbildung 9.19:
XML-Datei im
Internet Explorer

Abbildung 9.19 zeigt das Ergebnis. Nun werden Sie etwas enttäuscht sein. Praktisch ist nur der Quelltext zu sehen! Aber in welcher Form die Darstellung erfolgt, überrascht schon. Denn die Tags sind farbig gekennzeichnet – der IE hat offensichtlich die Tags erkannt und versucht zu interpretieren. Mehr geht an dieser Stelle nicht, denn woher soll der Browser wissen, was Sie sich bei dem Tag <AUSGABE> gedacht haben?

An dieser Stelle sollten Sie ein wenig mit dem Browser spielen. Versuchen Sie, das schließende Tag falsch zu schreiben oder auch Kleinbuchstaben zu verwenden. Sie werden bemerken, dass XML keineswegs so tolerant mit den Tags umgeht wie HTML – obwohl das Tag <AUSGABE> gänzlich unbekannt ist.

Der Internet Explorer verwendet einen eigenen XML-Parser, MSXML. Der im Folgenden von PHP genutzte Parser stammt von James Clark und zeigt im Detail andere Reaktionen.



Hinweis

Es ist nun an der Zeit, dem Tag Leben einzuhauchen. Im Prolog – dem einleitenden Teil eines XML-Dokuments – wird das Tag nun definiert:

Listing 9.28:
Einfache eingebettete
DTD [simple2.xml](#)

```
<?xml version="1.0" ?>
<!DOCTYPE FirstXML [
  <!ELEMENT AUSGABE (#PCDATA)>
]>
<AUSGABE>Hallo XML!</AUSGABE>
```

Die Reaktion bei dieser Version verwundert diesmal noch mehr. Es hat sich praktisch nicht viel geändert. Der Browser hat lediglich die Definition – konkret die Document-Type-Definition DTD – teilweise unterdrückt. Um noch etwas genauer erkennen zu können, welches Element wie definiert wird, sehen Sie sich die folgende Version an:

Listing 9.29:
DTD und XML-
Daten kombiniert in
[simple3.xml](#)

```
<?xml version="1.0" ?>
<!DOCTYPE FirstXML [
  <!ELEMENT AUSGABE (ANZEIGE)>
  <!ELEMENT ANZEIGE (#PCDATA)>
]>
<AUSGABE>
  <ANZEIGE>Jetzt wird's spannend!</ANZEIGE>
</AUSGABE>
```

Wie leicht zu erkennen ist, verwendet die DTD eine eigene Definitionssprache. Es ist nicht einsichtig, warum dies nicht im XML selbst erfolgt. Dieser Mangel wurde bereits erkannt und »XML-Schema« entwickelt, eine Definitionssprache für XML-Sprachen. DTDs sind dennoch weit verbreitet. Es kann aber als relativ sicher angesehen werden, dass auf mittelfristige Sicht die Norm »XML-Schema« das Rennen gewinnt.

9.6.2 Auswahl des XML-Parsers

PHP wird derzeit mit zwei Parsern geliefert: *Expat* und *Sablotron*. Beide stehen auch für Windows zur Verfügung. Bei Providern sieht es nicht so gut aus, meist steht, wenn überhaupt, nur *Expat* zur Verfügung. Deshalb konzentriert sich dieses Kapitel auch die entsprechenden XML-Funktionen dieses Moduls.

Expat-XML-Parser

James Clark's XML-Erweiterung EXPAT

Die Erweiterung nutzt *Expat* zur Darstellung der Funktionalität von XML in PHP. Sie können sich bei eigenen Forschungen nach Sinn und Anwendung von XML auch auf den Seiten von James Clark umsehen. Realisiert wird ein XML-Parser, aber kein Validator. Unterstützt werden die folgenden Zeichensätze:

- US-ASCII (Standard ASCII ohne Sonderzeichen)
- ISO-8859-1 (Erweiterter Satz mit westeuropäischen Zeichen)
- UTF-8 (Umfangreicher Zeichensatz mit asiatischen Zeichen)

Der Zeichensatz UTF-16 wird explizit nicht unterstützt. Dies betrifft nur Anwender spezieller asiatischer Sprachen. Sie können XML-Parser erstellen, diesen dann Ereignisbehandlungsroutinen zuordnen und sie mit Parametern steuern.

Sablotron Parser

Im Gegensatz zu Expat funktioniert Sablotron anders. Anstatt mit Schleifen durch die XML-Datei zu laufen, werden hier beim Auftreten bestimmter Elemente Ereignisse ausgelöst. Dies ist elegant, wenn komplexe Probleme zu lösen sind; für Anfänger aber unter Umständen schwerer zu programmieren, weil die zugrunde liegende Logik nicht sofort erkennbar wird.

9.6.3 Die Funktionsdeklarationen

Folgende Funktionen definieren eigene Routinen, um bei der Verarbeitung von XML-Dokumenten auf bestimmte Daten zu reagieren. Dabei wird eine weitere – benutzerdefinierte – Funktion namentlich angegeben. Diese sind also letztlich selbst für die Verarbeitung zuständig.

- `xml_set_element_handler`

Die hier definierte Funktion wird ausgeführt, wenn der Parser ein XML-Element erreicht oder eines verlässt. Start- und End-Tags werden getrennt behandelt.

- `xml_set_character_data_handler`

Diese Funktion behandelt alle Zeichen, die nicht als Tag gelten. Das betrifft auch Whitespaces.

- `xml_set_processing_instruction_handler`

Hiermit werden die einleitenden Tags, wie `<?`, spezifiziert.

- `xml_set_default_handler`

Diese Funktion wird gestartet, wenn sich keine andere für ein Ereignis zuständig fühlt.

- `xml_set_unparsed_entity_decl_handler`

Diese Funktion behandelt nicht zu parsende Einheiten (NDATA).

- `xml_set_notation_decl_handler`

Eine Notations-Deklaration wird hiermit behandelt.

- `xml_set_external_entity_ref_handler`

Wenn eine externe Referenz gefunden wird, ist diese Funktion zuständig.

**Spezielle
Eigenschaften der
XML-Funktionen**

Die XML-Funktionen haben einige allgemeine Eigenschaften, die Sie bei der Programmierung berücksichtigen müssen.

Umwandlung der Schreibweise**Case Folding**

Die als »Case Folding²⁸« bezeichnete Maßnahme bezieht sich auf die Anforderung der Ereignisbehandlungsroutinen, die zu untersuchen- den Tags in Großbuchstaben zu erhalten. Entsprechend werden alle XML-Elemente, die gesendet werden, zuvor in Großbuchstaben ge- wandelt. Der Vorgang kann allerdings mit den beiden Funktionen `xml_parser_get_option` und `xml_parser_set_option` kontrolliert wer- den, falls ein anderes Verhalten wünschenswert ist.

Fehlercodes

Die Fehlercodes, die von der Funktion `xml_parse` ausgegeben wer- den, liegen als Konstante vor. Tabelle 9.1 zeigt die Fehlercodes auf einen Blick.

Tabelle 9.1:
XML-Fehlercodes

Fehlercode

```
XML_ERROR_NONE
XML_ERROR_NO_MEMORY
XML_ERROR_SYNTAX
XML_ERROR_NO_ELEMENTS
XML_ERROR_INVALID_TOKEN
XML_ERROR_UNCLOSED_TOKEN
XML_ERROR_PARTIAL_CHAR
XML_ERROR_TAG_MISMATCH
XML_ERROR_DUPLICATE_ATTRIBUTE
XML_ERROR_JUNK_AFTER_DOC_ELEMENT
XML_ERROR_PARAM_ENTITY_REF
XML_ERROR_UNDEFINED_ENTITY
XML_ERROR_RECURSIVE_ENTITY_REF
XML_ERROR_ASYNC_ENTITY
XML_ERROR_BAD_CHAR_REF
```

²⁸ Die Übersetzung wäre nur schwer in ein gängiges deutsches Wort zu pres- sen. Da auch in der Literatur kaum eine deutsche Entsprechung verwendet wird, wurde hier der englische Begriff belassen.

Fehlercode

```
XML_ERROR_BINARY_ENTITY_REF
XML_ERROR_ATTRIBUTE_EXTERNAL_ENTITY_REF
XML_ERROR_MISPLACED_XML_PI
XML_ERROR_UNKNOWN_ENCODING
XML_ERROR_INCORRECT_ENCODING
XML_ERROR_UNCLOSED_CDATA_SECTION
XML_ERROR_EXTERNAL_ENTITY_HANDLING
```

Zeichenkodierung

PHP kodiert Zeichen intern immer im Unicode-Format UTF-8 (8 bis 21 Bits in bis zu 4 Bytes). Die Kodierung des Quelltextes muss vor dem Parsen festgelegt werden. Die alternativen Zeichensätze US-ASCII und ISO-8859-1 sind 8-Bit-Zeichensätze.

Eine weitere Einstellung ist nach dem Parsen der Tags möglich. Diese als Zielkodierung bezeichnete Wahl des Zeichensatzes erfolgt, wenn die untersuchten Elemente an die Ereignisbehandlungsroutinen geleitet werden. Zeichen, die nicht abgebildet werden können, werden durch ein Fragezeichen ersetzt.

9.6.4 Funktionsübersicht XML-Funktionen

Die folgenden Funktionsübersicht zeigt alle XML-Funktionen auf einen Blick. Mehr Informationen finden Sie auch in der Referenz. Die Liste der Deklarationsfunktionen wurden bereits in ➡ Abschnitt 9.6.3 *Die Funktionsdeklarationen* auf Seite 701 gezeigt.

- `xml_parser_create`
Erzeugt eine neue Instanz des XML-Parsers und gibt ein Handle zurück, der von anderen Funktionen benötigt wird.
- `xml_parse`
Startet das Parsen eines Dokuments.
- `xml_get_error_code`
Diese Funktion zeigt Fehlernummern an.
- `xml_error_string`
Hiermit werden Fehlermeldungen angezeigt.

- `xml_get_current_line_number`
Gibt die aktuell vom Parser untersuchte Zeile des XML-Dokuments aus.
- `xml_get_current_column_number`
Gibt die aktuell vom Parser untersuchte Spalte des XML-Dokuments aus.
- `xml_get_current_byte_index`
Gibt den aktuellen Byte-Index des XML-Dokuments aus.
- `xml_parser_free`
Gibt einen Parser wieder frei.
- `xml_parser_set_option`
Setzt verschiedene Optionen.
- `xml_parser_get_option`
Ermittelt die aktuellen Werte der Optionen.
- `utf8_decode`
Diese Funktion konvertiert eine UTF-8-Zeichenkette nach ISO-8859-1.
- `utf8_encode`
Hiermit wird eine ISO-8859-1-Zeichenkette nach UTF-8 konvertiert.

9.6.5 XML-Anwendungen in der Praxis

Die folgenden Beispiele zeigen, wie Sie selbst XML-Parser schreiben und damit Zugriff auf XML-Daten erlangen.

Der einfachste Parser

Das folgende Beispiel zeigt einen einfachen Parser, der nur Tags verarbeitet. Attribute und Daten innerhalb der Container werden nicht beachtet.

*Listing 9.30:
Beispiel xml_simple:
Einfachste Form
eines XML-Parsers*

```
<?php
$file = "file.xml";
$depth = 0;

function startElement($parser, $name, $attrs) {
    global $depth;
    echo str_repeat("&nbsp;", $depth * 4);
    echo "&lt;$name&gt;<br>";
}
```

```

    $depth++;
}

function endElement($parser, $name) {
    global $depth;
    $depth--;
    echo str_repeat("&nbsp;", $depth * 4);
    echo "&lt;/$name&gt;<br>";
}

$xml_parser = xml_parser_create();
xml_set_element_handler($xml_parser, "startElement",
    "endElement");
if (!$fp = fopen($file, "r")) {
    die("Kann XML-Datei nicht öffnen");
}
while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML-Fehler: %s auf Zeile %d",
            xml_error_string(xml_get_error_code($xml_parser)),
            xml_get_current_line_number($xml_parser)));
    }
}
xml_parser_free($xml_parser);
?>

```

Zuerst wird ein neuer Parser erzeugt, die Referenz *\$xml_parser* wird in allen anderen Funktionen zum Zugriff genutzt: **Wie es funktioniert**

```
$xml_parser = xml_parser_create();
```

Dann werden die Ereignisbehandlungsroutinen zugewiesen. Im einfachsten Fall werden Start- und End-Tags behandelt. Als Parameter werden die (willkürlich benannten) Funktionsnamen angegeben:

```
xml_set_element_handler($xml_parser,
    "startElement",
    "endElement");
```

Das Einlesen der XML-Datei erfolgt mit der übliche Schleife. Die Interpretation erfolgt zeilenweise mit der Anweisung

```
xml_parse($xml_parser, $data, feof($fp))
```

Die Ausgabe Richtung Browser erfolgt automatisch. Zum Schluss wird der Parser wieder zerstört:

```
xml_parser_free($xml_parser);
```

Die eigentliche Funktionalität liegt also in den Ereignisbehandlungsroutinen, die hier durch die Funktionen *startElement* und *endElement* gebildet werden. Das Beispiel rückt einfach jedes öffnende Element ein und entsprechend beim schließenden Element wieder aus. Die Anzahl der Einrückungen wird in der globalen Variablen *\$depth* gespeichert.

Wird ein öffnendes Element gefunden, wird eine Anzahl Leerzeichen ausgegeben:

```
echo str_repeat("&nbsp;", $depth * 4);
```

Anschließend wird für die nächste Stufe die Tiefe erhöht:

```
$depth++;
```

Einfacher ist die Darstellung der Behandlung für die schließenden Tags. Hier wird nur die Einrückung wieder zurückgenommen:

```
$depth--;
```

Das eigentliche XML-Dokument, das hier zur Anzeige gebracht wird, bestimmt die Variable *\$file*. Wenn als Datenbasis die Datei aus Listing 9.29 verwendet wird, ist folgende Ausgabe zu sehen:

Abbildung 9.20:
Anzeige der Struktur
einer XML-Datei

```
<AUSGABE>
  <ANZEIGE>
</ANZEIGE>
</AUSGABE>
```

Externe Entitäten

Im Gegensatz zum vorangegangenen Beispiel wird hier der Code konkret untersucht und durch farbige Hervorhebung die Reaktion des Parsers kenntlich gemacht. Das XML-Dokument wird wieder in der Variablen *\$file* bestimmt.

Listing 9.31:
Komplexer XML-
Parser: *xml_complex*

```
<?php
$file = "data.xml";

function indent($addsub = 0) {
    static $depth;
    if ($addsub < 0) {
        $indent = str_repeat("&nbsp;", $depth * 4);
        $depth += $addsub;
        return $indent;
    } else {
        $depth += $addsub;
        $indent = str_repeat("&nbsp;", $depth * 4);
        return $indent;
    }
}

function startElement($parser, $name, $attr) {
    echo indent(1);
    print "&lt;<span class=tag>$name</span>";
    if (is_array($attr)) {
        foreach ($attr as $k => $v) {
            echo " <span class=attr>$k</span>=";
            echo "'<span class=parm>$v</span>'";
        }
    }
}
```

```
    print "&gt;<br>";
}

function endElement($parser, $name) {
    echo indent(-1);
    print "&lt;/span class=tag>$name</span>&gt;<br>";
}

function characterData($parser, $data) {
    $data = chop($data);
    $indent = indent();
    print strlen($data) > 0 ? "$indent<b>$data</b><br>" : NULL;
}

function PIHandler($parser, $target, $data) {
    switch (strtolower($target)) {
        case "php":
            eval($data);
            break;
        default:
            echo htmlspecialchars($data);
    }
}

function defaultHandler($parser, $data) {
    $data = chop($data);
    echo indent();
    if (preg_match("/^&.*;$|U", $data)) {
        echo '<u>' . htmlspecialchars($data) . '</u><br>';
    } elseif (strlen($data) > 0) {
        echo htmlspecialchars($data);
        echo '<br>';
    }
}

function externalEntityRefHandler($parser, $openEntityNames,
                                   $base, $systemId, $publicId) {
    if ($systemId) {
        if (!list($parser, $fp) = new_xml_parser($systemId)) {
            printf("Kann Entity %s in Zeile %s nicht öffnen<br>\n",
                $openEntityNames, $systemId);
            return false;
        }
        while ($data = fread($fp, 4096)) {
            if (!xml_parse($parser, $data, feof($fp))) {
                printf("XM-Fehler: %s in Zeile %d\n",
                    während der Verarbeitung von %s<br>\n",
                    xml_error_string(xml_get_error_code($parser)),
                    xml_get_current_line_number($parser),
                    $openEntityNames);
                xml_parser_free($parser);
            }
        }
    }
}
```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

```

        return false;
    }
}
xml_parser_free($parser);
return true;
}
return false;
}

function new_xml_parser($file) {

    $xml_parser = xml_parser_create();
    xml_parser_set_option($xml_parser,
        XML_OPTION_CASE_FOLDING, 0);
    xml_set_element_handler($xml_parser, "startElement",
        "endElement");
    xml_set_character_data_handler($xml_parser, "characterData");
    xml_set_processing_instruction_handler($xml_parser,
        "PIHandler");
    xml_set_default_handler($xml_parser, "defaultHandler");
    xml_set_external_entity_ref_handler($xml_parser,
        "externalEntityRefHandler");

    if (!($fp = @fopen($file, "r"))) {
        return false;
    }
    if (!is_array($parser_file)) {
        settype($parser_file, "array");
    }
    $parser_file[$xml_parser] = $file;
    return array($xml_parser, $fp);
}

if (!(list($xml_parser, $fp) = new_xml_parser($file))) {
    die("Kann XML-Datei nicht öffnen");
}

print "<code>";
while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML-Fehler: %s in Zeile %d<br>\n",
            xml_error_string(xml_get_error_code($xml_parser)),
            xml_get_current_line_number($xml_parser)));
    }
}
print "</code>";
print "parse complete\n";
xml_parser_free($xml_parser);
?>

```

Wie es funktioniert Zur Anwendung sollten Sie zuerst die verwendete XML-Datei »data.xml« betrachten:

```

<?xml version='1.0'?>
<!DOCTYPE chapter SYSTEM "/just/a/test.dtd" [
  <!ENTITY plainEntity "FOO entity">
  <!ENTITY systemEntity SYSTEM "external.xml">
]>
<chapter>
  <TITLE>Title &plainEntity;</TITLE>
  <para>
    <informaltable>
      <tgroup cols="3">
        <tbody>
          <row>
            <entry>a1</entry>
            <entry morerows="1">b1</entry>
            <entry>c1</entry>
          </row>
          <row>
            <entry>a2</entry>
            <entry>c2</entry>
          </row>
          <row>
            <entry>a3</entry>
            <entry>b3</entry>
            <entry>c3</entry>
          </row>
        </tbody>
      </tgroup>
    </informaltable>
  </para>
  &systemEntity;
  <sect1 id="about">
    <title>Über diesen Text</title>
    <para>
      <!-- Dies ist ein Kommentar -->
      <?php echo 'Hallo! Die ist PHP Version ' .phpversion(); ?>
    </para>
  </sect1>
</chapter>

```

Listing 9.32:
Beispieldatei
data.xml für die
Parser in Listing
9.30 und Listing 9.31

Das Erzeugen des Parsers ist hier etwas komplexer, es werden auch Prozessbehandlungsroutinen integriert. Die Definition der Behandlungsroutinen erfolgt innerhalb der Funktion `new_xml_parser`:

Arbeitsweise des Parsers

```

$xml_parser = xml_parser_create();
xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, 1);
xml_set_element_handler($xml_parser,
    "startElement",
    "endElement");
xml_set_character_data_handler($xml_parser, "characterData");
xml_set_processing_instruction_handler($xml_parser, "PIHandler");
xml_set_default_handler($xml_parser, "defaultHandler");

```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

```
xml_set_external_entity_ref_handler($xml_parser,
                                   "externalEntityRefHandler");
```

Die Behandlung der Start- und End-Tags wurde bereits im letzten Beispiel gezeigt. Die Erweiterung um die »Colorierung« dürfte sich selbst erklären. Spannender ist die Prozessbehandlung mit der Funktion `PIHandler`. Hier wird zuerst ausgewertet, ob PHP-Code vorhanden ist (die Sicherheitsmaßnahme »trusted file« wird nachfolgend erläutert). Zuerst wird das (aus XML-Sicht) spezielle Tag `<?php` erkannt:

```
switch (strtolower($target)) {
    case "php":
```

Ist es zulässiger PHP-Code, erfolgt die Abarbeitung mit der Funktion `eval` (das hat natürlich nichts mit XML zu tun):

```
    eval($data);
```

Ersatzweise wird der Code einfach ausgegeben, ohne abgearbeitet zu werden:

```
    echo htmlspecialchars($data);
```

Sehr einfach ist auch die Behandlung aller Zeichen, die nicht vom Parser behandelt werden. Die Funktion `characterData` schreibt diese einfach fett:

```
print strlen($data) > 0 ? "$indent<b>$data</b><br>" : NULL;
```

Die externen Referenzen müssen getrennt von der Basisdatei behandelt werden. Tritt eine solche Referenz auf, ist die Funktion `externalEntityRefHandler` zuständig. Hier erfolgt der Start eines weiteren Parsers:

```
if (!list($parser, $fp) = new_xml_parser($systemId)) {
```

Die Aufforderung zum Lesen eines weiteren Dokuments finden Sie im XML-Code der Beispieldatei `DATA.XML`:

```
<!ENTITY systemEntity SYSTEM "external.xml">
```

Entsprechend muss dieses Dokument im Zugriff des Parsers sein:

```
<?xml version="1.0"?>
<!DOCTYPE foo [
<!ENTITY testEnt "test entity">
]>
<foo>
  <element attrib="value"/>
  &testEnt;
  <?php
    $check = phpversion();
    print "Hier wurde PHP-Code ausgef&uuml;hrt. PHP-Version:
        $check";
```

Listing 9.33:
Datei external.xml
zum vorange-
gangenen Beispiel

```
?>  
</foo>
```

Im gesamten Skript wird eine Einrückung mitgeführt, realisiert über die Funktion *indent()*. Damit wird die Struktur des Dokuments bei der Anzeige besser lesbar.

Mit Hilfe dieses Skripts können Sie praktisch gezielt auf alle Elemente der XML-Datei zugreifen. Allerdings wird immer das gesamte Dokument – von oben nach unten – verarbeitet. Eine Analyse von Teilen bzw. die gezielte Extraktion von Daten ist nicht ohne weiteres möglich. Praktisch können Sie derartige Methoden verwenden, um fremde Daten in ein eigenes Format oder eine relationale Datenbank zu überführen. Für die Navigation innerhalb von XML wurde XPath entwickelt, was den Einsatz anderer Module voraussetzt.

Die Ausgabe sehen Sie in Abbildung 9.21 auf Seite 712.

9.6.6 Das XML-Projekt

Echte Praxisnähe konnte man in den letzten beiden Beispielen noch nicht erkennen – es ging auch eher um die Vorführung der Funktionen. Das hier gezeigte Projekt ist eine Erweiterung des Projekts *phpHoo* aus Kapitel 7. Dort wird eine einfache Suchmaschine implementiert, die Kategorie und Daten wie bei Yahoo! darstellt.

Einführung

Eine eigene Suchmaschine ist ein spannendes Projekt. Richtig sinnvoll wird es aber erst, wenn genug Daten vorhanden sind. Auch hierfür gibt es glücklicherweise ein Open Source Projekt: DMOZ. Unter der Adresse <http://dmoz.org> finden Sie alle Informationen und eine funktionierende Umgebung. Die Daten können kostenfrei lizenziert werden, lediglich ein Link auf die Quelle ist erforderlich.

Das gesamte Projekt besteht aus zwei Teilen:

1. Eine leicht gewandelte Version von *phpHoo*
2. Ein XML-Parser zum Import der Verzeichnisdaten

Die XML-Daten selbst liegen in zwei Dateien vor, die zum einen die Struktur und zum anderen die Links enthalten. Der Parser durchsucht diese Dateien und überträgt die Inhalte in eine MySQL-Datenbank, auf die die Anzeigefunktionen zugreifen.

Das Projekt

RDF

Abbildung 9.21:
Analyse eines
komplexen XML-
Dokuments (Ausgabe
des Skripts aus
Listing 9.31)

XML Daten analysieren

```
<?xml version='1.0'?>
<!DOCTYPE
chapter
SYSTEM
"/just/a/test.dtd"
[
<!ENTITY
plainEntity
"FOO entity"
>
<!ENTITY
systemEntity
SYSTEM
"external.xml"
>
]
>

<chapter>
  <TITLE>
    Title
    plainEntity;
  </TITLE>
  <para>
    <informaltable>
      <tgroup cols='3'>
        <tbody>
          <row>
            <entry>
              a1
            </entry>
            <entry morerows='1'>
              b1
            </entry>
            <entry>
              c1
            </entry>
          </row>
          <row>
            <entry>
              a2
            </entry>
            <entry>
              c2
            </entry>
          </row>
          <row>
            <entry>
              a3
            </entry>
            <entry>
              b3
            </entry>
            <entry>
              c3
            </entry>
          </row>
        </tbody>
      </tgroup>
    </informaltable>
  </para>
  <?xml version="1.0"?>
  <!DOCTYPE
foo
[
<!ENTITY
testEnt
"test entity"
>
]
>
  <foo>
    <element attrib='value'>
      </element>
    <testEnt>
      Hier wurde PHP-Code ausgeführt. PHP-Version: 4.0.7-dev
    </testEnt>
    <sect1 id='about'>
      <title>
        Über diesen Text
      </title>
      <para>
        <!-- Dies ist ein Kommentar -->
      </para>
      Hallo! Dies ist PHP Version 4.0.7-dev
    </sect1>
  </foo>
</chapter>
```

Parser beendet

Die Datenbank entwerfen

Zuerst muss eine Datenbank entworfen werden. Die beiden folgenden Definitionen zeigen, wie Struktur und Inhalt abgelegt werden. Zuerst die Struktur des Verzeichnisses, die in der Tabelle *topic* gespeichert wird:

```
CREATE TABLE topic (
  id int(11) NOT NULL auto_increment,
  topic_id int(11) NOT NULL,
  topic_parent int(11) NOT NULL,
  topic_title varchar(255) NOT NULL,
  topic_level tinyint(4) NOT NULL,
  topic_desc text,
  KEY topic_parent (topic_parent),
  PRIMARY KEY (id),
  KEY topic_id (topic_id),
  KEY topic_title (topic_title)
);
```

*Listing 9.34:
SQL-Anweisung
zum Erstellen der
Tabelle »topic«
([topic.sql](#))*

Die Struktur realisiert einen B-Baum, bei dem jedes Element (*topic_id*) einen Verweis auf ein übergeordnetes Element enthält (*topic_parent*) und zugleich das Niveau oder die Ebene innerhalb des Baumes angegeben wird (*topic_level*).

Die Indizes sind unbedingt notwendig, da das Verzeichnis sehr groß ist und ohne Indizierung unmöglich durchsucht werden kann. Der Inhalt wird in einer zweiten Tabelle *content* abgelegt:

```
CREATE TABLE content (
  topic_id int(11) NOT NULL,
  content_page varchar(255) NOT NULL,
  content_title varchar(255),
  content_desc text,
  id int(11) NOT NULL auto_increment,
  PRIMARY KEY (id),
  KEY topic_id (topic_id)
);
```

*Listing 9.35:
Links werden in der
Tabelle »content«
gespeichert
([content.sql](#))*

Das Feld *topic_id* verweist auf *topic_id* in der Tabelle *topic*.

Die Struktur der Datenquelle

Die Datenquelle ist eine RDF-Datei. RDF steht für »Resource Description Framework«. Darunter wird ein XML-Format verstanden, das vor allem zur Beschreibung von webbasierten Metadaten dient. Die Anwendung als Datenspeicher für einen Katalog ist nur eine mögliche Anwendung. Eine vollständige Definition und viele Informationen finden Sie unter:

<http://www.w3.org/RDF>

Das folgende Listing zeigt, wie die erste Ebene definiert wird:

Struktur

Listing 9.36:
Definition der ersten
Katalogebene
(structure.xml)

```
<RDF xmlns:r="http://www.w3.org/TR/RDF/"
      xmlns:d="http://purl.org/dc/elements/1.0/"
      xmlns="http://directory.mozilla.org/rdf">

  <Topic r:id="Top">
    <tag catid="1"/>
    <d:Title>Top</d:Title>
    <narrow r:resource="Top/Arts"/>
    <narrow r:resource="Top/Business"/>
    <narrow r:resource="Top/Computers"/>
    <narrow r:resource="Top/Games"/>
    <narrow r:resource="Top/Health"/>
    <narrow r:resource="Top/Home"/>
    <narrow r:resource="Top/News"/>
    <narrow r:resource="Top/Recreation"/>
    <narrow r:resource="Top/Reference"/>
    <narrow r:resource="Top/Regional"/>
    <narrow r:resource="Top/Science"/>
    <narrow r:resource="Top/Shopping"/>
    <narrow r:resource="Top/Society"/>
    <narrow r:resource="Top/Sports"/>
    <narrow r:resource="Top/Test"/>
    <symbolic r:resource="Typography:Top/Computers/Fonts"/>
    <newsGroup r:resource="news:comp.misc"/>
  </Topic>
```

Struktur der Daten Jede Ebene wird mit dem Element `<Topic>` definiert. Das Attribut `r:id` enthält den Namen:

```
<Topic r:id="Top">
```

Jeder Eintrag enthält außerdem eine eindeutige Katalog-ID, die auch in der Datenbank genutzt wird:

```
<tag catid="1"/>
```

Dann folgt ein beschreibender Titel, der auch zur Anzeige verwendet wird:

```
<d:Title>Top</d:Title>
```

Es folgt nun die Liste der Elemente, die unterhalb dieser Ebene stehen:

```
<narrow r:resource="Top/Arts"/>
<narrow r:resource="Top/Business"/>
<narrow r:resource="Top/Computers"/>
```

Für das Projekt wird es nicht ausgewertet, für eigene Experimente bietet es sich aber an: Neben Links zu tieferen Ebenen werden weitere Verweise gebildet. `<symbolic>` definiert Querverweise (in Yahoo! werden diese durch das @-Zeichen kenntlich gemacht):

```
<symbolic r:resource="Typography:Top/Computers/Fonts"/>
```

Newsgroups, die keine weiteren Elemente enthalten können, werden ebenfalls direkt definiert:

```
<newsGroup r:resource="news:comp.misc"/>
```

Auch der Inhalt wird als XML-Datei abgebildet. Das folgende Listing zeigt einen Ausschnitt:

```
<RDF xmlns:r="http://www.w3.org/TR/RDF/"
      xmlns:d="http://purl.org/dc/elements/1.0/"
      xmlns="http://directory.mozilla.org/rdf">

  <Topic r:id="Top">
    <tag catid="1"/>
    <d:Title>Top</d:Title>
  </Topic>

  <Topic r:id="Top/Arts">
    <tag catid="2"/>
    <d:Title>Arts</d:Title>
    <link r:resource="http://www3.bc.sympatico.ca/GlassPage.html"/>
  </Topic>

  <ExternalPage about="http://www3.bc.sympatico.ca/GlassPage.html">
    <d:Title>John phillips Blown glass</d:Title>
    <d:Description>A small display of glass by...</d:Description>
  </ExternalPage>
```

*Listing 9.37:
Aufbau der XML-
Datei für den Inhalt:
[content.xml](#)*

Abgesehen vom Kopf (<RDF ...>) werden in der Datei zwei Elemente definiert: <Topic> und <ExternalPage>. Topic bildet die Verbindung zur Datei STRUCTURE.XML und den dort definierten Elementen. Schauen Sie sich den Aufbau genau an.

Struktur der Daten

Jeder Topic beginnt mit der Nennung des kompletten Pfades, der Position im Baum also:

```
<Topic r:id="Top/Arts">
```

Dann wird die Katalog-ID definiert, die später zur Zuordnung herangezogen wird:

```
<tag catid="2"/>
```

Der nächste Eintrag definiert den Titel des Topics:

```
<d:Title>Arts</d:Title>
```

Wenn in dem Topic Links sind, werden diese nachfolgend aufgelistet:

```
<link r:resource="http://www3.bc.sympatico.ca/GlassPage.html"/>
```

Im letzten Schritt wird das Element wieder geschlossen:

```
</Topic>
```

Wenn Links mit dem Tag `<link r:resource...>` definiert wurden, folgt dem Element `<Topic>` unmittelbar das Element `<ExternalPage>` mit einer genauen Beschreibung. Zuerst wird mit dem Attribut `about` die Verbindung zum Element `<link...>` hergestellt:

```
<ExternalPage about="http://www3.bc.sympatico.ca/GlassPage.html">
```

Dann wird der Titel des Links und eine Beschreibung definiert :

```
<d:Title>John phillips Blown glass</d:Title>
<d:Description>A small display of glass by...</d:Description>
```

Zuletzt wird das Element geschlossen:

```
</ExternalPage>
```

Der Parser

Nachfolgend wird der Parser gezeigt, der als Klasse definiert wurde. Das komplette Listing finden Sie auf der CD im Verzeichnis ANWENDUNGEN/PHPXMLHOO.

Zuerst wird die Datenquelle definiert. Passen Sie diesen Teil gegebenenfalls an Ihre Bedürfnisse an.

```
<?php
define(STRUCTURE, "../data/structure.xml");
define(CONTENT, "../data/content.xml");
```

Jetzt erfolgt die Definition der Klasse.

```
class import_dmoz {
```

Die Klasse beginnt mit der Definition der Eigenschaften. Die drei Variablen enthalten Handle auf den XML-Parser und die geöffneten Datendateien:

```
var $hdlParser;
var $hdlDataStructure;
var $hdlDataContent;
```

Zwei weitere Eigenschaften enthalten die Dateinamen der Datendateien:

```
var $strStructure;
var $strContent;
```

Da die Struktur der Daten in der Datenbank nicht der der XML-Dateien entspricht, müssen einige Werte zwischengespeichert werden. Die folgenden Eigenschaften speichern diese Werte:

```
var $strTopic;           # Topic
var $strTitle;           # Name des Topic oder Links
var $strDescription;     # Beschreibender Text
var $intCatid;           # Katalog-ID
```



Listing 9.38:
phpHoo, das XML-
Projekt (wird mit
Kommentaren
fortgesetzt)

↓

↓

↓

↓

```
var $strPage;      # Link zur Seite (Content)
var $strNarrow;    # Verweis auf die Content-Datei
```

Um die Funktionen des Parsers zu steuern, werden weitere Eigenschaften eingesetzt:

```
var $strNextElement; # Nächstes einzulesendes Element
var $bInUpdate;      # Update in der zweiten Phase
var $bInContent;     # TRUE, wenn Content-Import
```

Die erste Methode ist der Constructor. Diese Methode wird aufgerufen, wenn eine Instanz der Klasse angelegt wird. Erzeugt wird ein Handle auf einen neuen XML-Parser. Anschließend wird dieser Parser innerhalb des Objekts sichtbar gemacht (`xml_set_object`). Standardmäßig kann ein Parser nicht innerhalb einer Klasse erreicht werden:

```
function import_dmoz() {
    $this->hdlParser = xml_parser_create("UTF-8");
    xml_set_object($this->hdlParser, &$this);
}
```

Die Methode `chrData` liest die Daten innerhalb der `<TOPIC>`- und `<NARROW>`-Elemente. Der Aufruf erfolgt in jedem Element, das Attribute enthält. Wohin die Daten gespeichert werden, entscheidet die Eigenschaft `strNextElement`. Mit Hilfe der switch-Anweisung werden die Daten der richtigen Eigenschaft zugewiesen. Beachten Sie außerdem die Umwandlung der UTF-8-Kodierung, ohne die die Umlaute nicht korrekt erscheinen würden:

```
function charData($hdlParser, $strData) {
    if (strlen(trim($strData)) > 0) {
        switch($this->strNextElement) {
            case "CATID":
                $this->intCatid = (int) $strData;
                break;
            case "D:DESCRIPTION":
                $this->strDescription .=
                    (string) addslashes(utf8_decode($strData));
                break;
            case "D:TITLE":
                $this->strTitle =
                    (string) addslashes(utf8_decode($strData));
                break;
        } /* end switch */
    } /* end if */
} /* end function charData */
```

Die Methode `startElement` wird mit jedem öffnenden Tag der XML-Datei aufgerufen. Attribute werden als Array übergeben, da mehrere Attribute zulässig sind. Dieses Array ist assoziiert und als Schlüsselname wird der Name des Attributes eingesetzt:

↓

```
function startElement($hdlParser, $strName, $arrAttribute) {
    switch($strName) {
```

Beginnt ein neuer Topic, wird der Name dem Attribut `r:id` entnommen und in der Eigenschaft `strTopic` gespeichert:

↓

```
        case "TOPIC": # Beginn eines neuen Topic
            $this->strTopic =
                addslashes(utf8_decode($arrAttribute["R:ID"]));
            break;
```

Innerhalb des Topics werden die untergeordneten Elemente definiert. Die Zuordnung kann erst erfolgen, wenn zuvor alle Topics erfasst wurden. Dieser Teil wird erst beim zweiten Durchlauf der Datei verwendet:

↓

```
        case "NARROW": # Beginn eines Verweises, nur im
                        # zweiten Durchgang (blnUpdate = TRUE)
            if ($this->blnUpdate == TRUE) {
                $this->strNarrow =
                    addslashes(utf8_decode($arrAttribute["R:RESOURCE"]));
```

Die Ebene innerhalb der Struktur wird äußerst unspektakulär ermittelt. Die Ebenen sind durch `/`-Zeichen getrennt. Der folgende Code zählt einfach diese Anzahl:

↓

```
                $arrNarrow = explode("/", $this->strNarrow);
                $intLevel = count($arrNarrow) - 1;
                $strTopic = $arrNarrow[$intLevel];
```

Jetzt sind alle Daten vorhanden und die Zuordnung kann aktualisiert werden:

↓

```
                $strSQL = "UPDATE topic ";
                $strSQL .= "SET topic_parent=$this->intCatid WHERE";
                $strSQL .= "topic_title = '$strTopic' ";
                $strSQL .= "AND topic_level = $intLevel";
                my_query ($strSQL, FALSE);
            } /* end if */
            break;
```

Handelt es sich um Inhalt (aus der Datei `CONTENT.XML`), kann das Element `<ExternalPage>` auftreten. Hier ist das Attribut `about` interessant:

↓

```
        case "EXTERNALPAGE":
            $this->strPage = addslashes($arrAttribute["ABOUT"]);
            break;
```

In allen anderen Fällen wird der Name des Elements gespeichert, damit `charData` auf den Inhalt zugreifen kann:

↓

```
        default:
            $this->strNextElement = $strName;
```

```
    } /* end switch */
  } /* end function startElement */
```

Auch das Ende eines Elements ist von Bedeutung. An dieser Stelle sollten alle Daten vollständig sein. Folgerichtig werden hier die INSERT-Anweisungen zu finden sein. Die Funktion wird aufgerufen, wenn ein Element geschlossen wird:

```
function endElement($hdlParser, $strName) {
  switch($strName) {
```

Topic ist nur im ersten Durchlauf interessant, kommt aber auch in der zweiten Datei vor, dazu die if-Anweisung am Anfang:

```
    case "TOPIC":          # Ende des Topics
      if ($this->blnUpdate == FALSE
          and $this->blnContent == FALSE) {
```

In bekannter Weise wird wieder die Ebene ermittelt:

```
      $intLevel = count(explode("/", $this->strTopic))-1;
```

Mit den inzwischen von den anderen Funktionen ermittelten Daten kann nun die INSERT-Anweisung aufgebaut werden:

```
      $strSQL = "INSERT INTO topic (topic_id,
                                   topic_parent,
                                   topic_title,
                                   topic_level,
                                   topic_desc) ";
      $strSQL .= "VALUES ($this->intCatid, 0,
                          '$this->strTitle', $intLevel,
                          '$this->strDescription')";
      my_query($strSQL, FALSE);
```

Eine Besonderheit stellt die Beschreibung dar. Da dieser Teil HTML-Tags enthalten kann, führt das zu Problemen mit dem Parser. Die *charData*-Funktion wird mehrfach aufgerufen. Damit trotzdem die komplette Beschreibung erfasst wird, sammelt die Eigenschaft *strDescription* diese. Am Ende muss der Wert gezielt gelöscht werden:

```
      unset($this->strDescription);
    } /* end if */
    break;
```

Beim Erfassen des Inhalts löst das Ende des Elements `<ExternalPage>` die INSERT-Anweisung aus:

```
    case "EXTERNALPAGE": # Ende der Seitenbeschreibung
      $strSQL = "INSERT INTO content (topic_id,
                                     content_page,
                                     content_title,
                                     content_desc) ";
      $strSQL .= "VALUES ($this->intCatid,
                          '$this->strPage',
```



```

        '$this->strTitle',
        '$this->strDescription');"
    my_query($strSQL, FALSE);
    unset($this->strDescription);
    break;
} /* end switch */
} /* end function endElement */

```

Leider ist die XML-Datei an einigen Stellen fehlerhaft. Deshalb wird eine Funktion implementiert, die das Ende des fehlerhaften Elements sucht und erst dann fortsetzt:

↓

```

function avoid_error($hdlFile) {
    $dummy = fgets($hdlFile, 4096);
    if ((strpos($dummy, "/Topic>") >= 1)
        or (strpos($dummy, "/ExternalPage>") >= 1)) {
        return FALSE;
    } else {
        return TRUE;
    } /* end if */
}

```

Jetzt folgen die eigentlichen Hauptfunktionen. *parse_xml_topics* liest die Datei STRUCTURE.XML und übernimmt die Daten in die Datenbank.

↓

```
function parse_xml_topics() {
```

Zuerst werden die Handler-Funktionen definiert, die zuvor bereits beschrieben wurden:

↓

```

    xml_set_element_handler($this->hdlParser, "startElement",
                           "endElement");
    xml_set_character_data_handler($this->hdlParser,
                                   "charData");

    $strStructure = STRUCTURE;
    $this->hdlDataStructure = fopen($strStructure, "r");
    // Einlesen aller Topics
    $this->bInUpdate = FALSE;
    $this->bInContent = FALSE;
    echo "<p>Beginne Import der Daten...</p>";

```

Der eigentliche Parser wird in einer *while*-Schleife aufgerufen, die die Datei zeilenweise durchläuft. Der Aufruf erfolgt in einer *if*-Anweisung. Die Funktion *xml_parse* gibt *FALSE* zurück, wenn ein Fehler auftrat:

↓

```

    while($strXMLData = fgets($this->hdlDataStructure, 4096)) {
        if(!xml_parse($this->hdlParser,
                     $strXMLData,
                     feof($this->hdlDataStructure))) {

```

Fehler führen nicht zum Abbruch, sondern nur zur Anzeige des Fehlers mit Hilfe der entsprechenden XML-Funktionen:

```
        printf("<br>XML Fehler: %s in Zeile %d",  
              xml_error_string(  
                xml_get_error_code($this->hdlParser),  
                xml_get_current_line_number($this->hdlParser));
```

↓

Anschließend wird der fehlerhafte Teil übersprungen:

```
        while($this->avoid_error($this->hdlDataStructure));  
    } /* end if */  
} /* end while */
```

↓

Dann wird die Quelldatei und der Parser geschlossen:

```
fclose($this->hdlDataStructure);  
echo "<p>Einlesen der Topics beendet...</p>";  
xml_parser_free($this->hdlParser);  
} /* end function parse_xml_topics */
```

↓

Fast identisch ist der Aufbau der Update-Funktion, die auf der Basis der bereits importierten Elemente die Verknüpfung zu den »Eltern-Elementen« herstellt:

```
function parse_xml_topics_update() {  
    xml_set_element_handler($this->hdlParser,  
                           "startElement", "endElement");  
    xml_set_character_data_handler($this->hdlParser,  
                                  "charData");  
    $strStructure = STRUCTURE;  
    $this->hdlDataStructure = fopen($strStructure, "r");  
    $this->blnUpdate = TRUE;  
    $this->blnContent = FALSE;  
    echo "<p>Beginne Update der Daten...</p>";  
    while($strXMLData = fgets($this->hdlDataStructure, 4096)) {  
        if(!xml_parse($this->hdlParser, $strXMLData,  
                      feof($this->hdlDataStructure))) {  
            printf("<br>XML Fehler: %s in Zeile %d",  
                  xml_error_string(  
                    xml_get_error_code($this->hdlParser),  
                    xml_get_current_line_number($this->hdlParser));  
            while($this->avoid_error($this->hdlDataStructure));  
        } /* end if */  
    } /* end while */  
    fclose($this->hdlDataStructure);  
    echo "<p>Update der Topics beendet...</p>";  
    xml_parser_free($this->hdlParser);  
} /* end function parse_xml_topics */
```

↓

Ebenso ist auch die Funktion aufgebaut, die den Inhalt importiert:

```
function parse_xml_content() {  
    $this->blnContent = TRUE;  
    xml_set_element_handler($this->hdlParser, "startElement",  
                           "endElement");  
    xml_set_character_data_handler($this->hdlParser,
```

↓

```

"charData");
$strContent = CONTENT;
$this->hdlDataContent = fopen($strContent, "r");
echo "<p>Beginne Import der Inhaltsdaten...</p>";
while($strXMLData = fgets($this->hdlDataContent, 4096)) {
    if(!xml_parse($this->hdlParser, $strXMLData,
        feof($this->hdlDataContent))) {
        printf("<br>XML Fehler: %s in Zeile %d",
            xml_error_string(
                xml_get_error_code($this->hdlParser)),
            xml_get_current_line_number($this->hdlParser));
        while($this->avoid_error($this->hdlDataContent));
    } /* end if */
} /* end while */
fclose($this->hdlDataContent);
echo "<p>Einlesen des Contents beendet...</p>";
xml_parser_free($this->hdlParser);
} /* end function parse_xml_content */

```

Die Klasse ist nun vollständig definiert und wird in den folgenden drei Skripten verwendet.

↓

```

} /* end class import_dmoz */
?>

```

Der Aufruf erfolgt in drei Skripten:

- IMPORT_TOPIC.PHP. Importiert die Topics.
- IMPORT_TOPIC_UPDATE.PHP. Stellt die Verknüpfungen her.
- IMPORT_CONTENT.PHP. Importiert den Inhalt.



Sie können alle drei Skripte bequem über das Skript IMPORT.PHP4 erreichen. Die Skripte rufen die Datei OPEN.INC.PHP4 auf, in der die bereits vorgestellte Klasse definiert wurde. Außerdem sind dort zwei Funktionen (hier nicht abgedruckt), die die Datenbanktabelle löschen und neu anlegen.

Listing 9.39:
Import der Topics:
import_topic

```

<?php
ob_end_flush();
echo "<p>Vorbereitung...</p>";
// Erzeugen der Tabelle, Loeschen der alten
create_table_topics();
// Import der Daten
$clsDmoz = new import_dmoz;
$clsDmoz->parse_xml_topics();
?>

```

Listing 9.40:
Verknüpfung der
Topics:
import_topic_update

```

<?php
ob_end_flush();
echo "<p>Vorbereitung...</p>";
// Import der Daten
$clsDmoz = new import_dmoz;

```

```
$clsDmoz->parse_xml_topics_update();
?>
```

```
<?php
ob_end_flush();
echo "<p>Vorbereitung...</p>";
// Erzeugen der Tabelle, Loeschen der alten
create_table_content();
// Import der Daten
$clsDmoz = new import_Dmoz;
$clsDmoz->parse_xml_content();
?>
```

Listing 9.41:
Import des Inhalts:
import_content

Nach dem Import der Daten können Sie das modifizierte phpHoo-Skript verwenden. Es liegt ebenfalls im Verzeichnis /ANWENDUNGEN/PHPXMLHOO auf der CD unter dem Namen MAIN.PHP4. Die folgenden Abbildungen zeigen, wie es fertig aussehen sollte. Auf eine Erläuterung soll hier verzichtet werden, die Änderungen sind eher kosmetischer Natur oder betreffen die Feldnamen der Datenbank, die hier an das XML-Projekt angepasst wurden.

Sie können den oben angezeigten Pfad nutzen, um direkt in eine der Kategorien zu springen. Die Suchanfrage kombiniert Wörter mit UND, durchsucht aber sowohl den Titel wie auch den Link und die Beschreibung.

Sie finden alle Skripte im Projekt XML-Projekt *phpHoo* auf der CD im Verzeichnis /ANWENDUNGEN/PHPXMLHOO.



Abbildung 9.22:
Anzeige des DMOZ-
Verzeichnisses mit
Navigationsleiste,
Liste der Kategorien
und Links

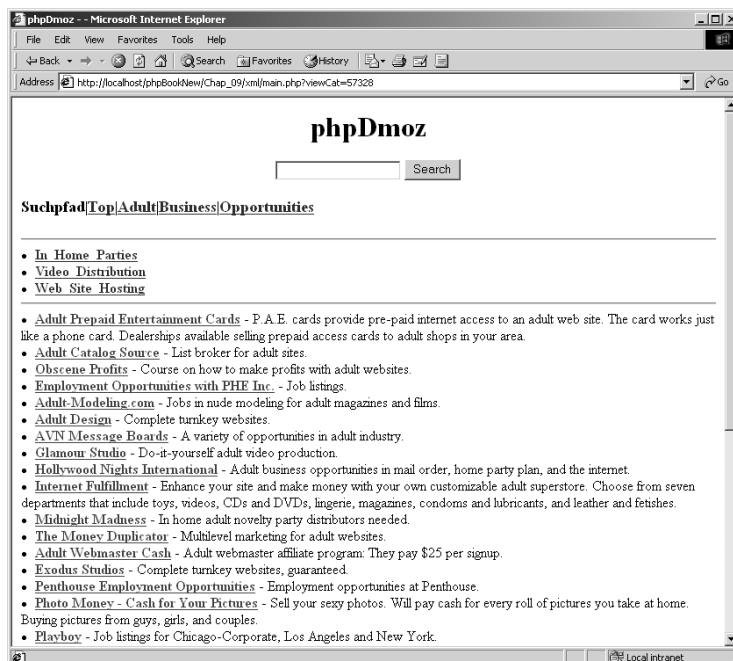
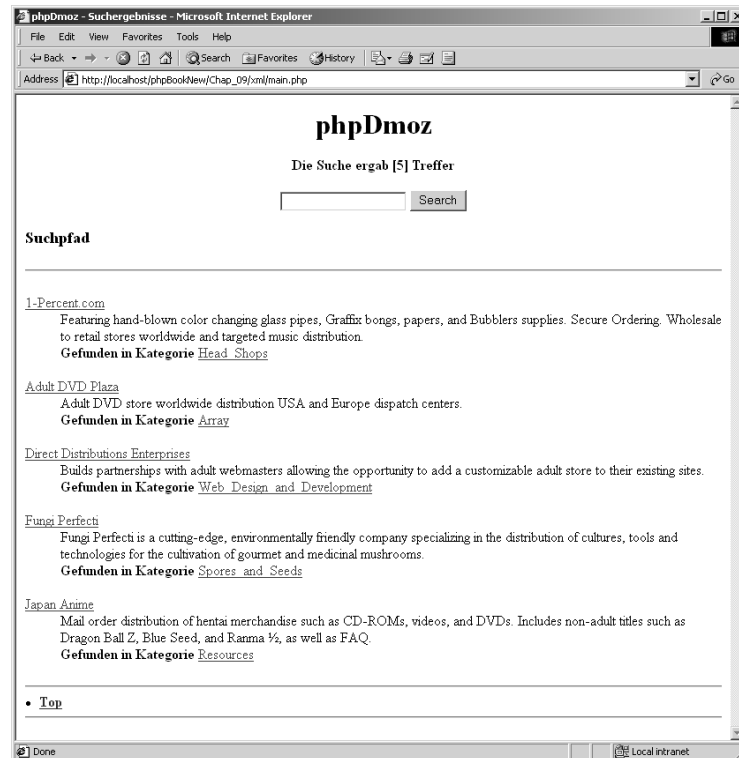


Abbildung 9.23:
Ergebnis einer
Suchanfrage mit
Links zu den Seiten
und zu den
Kategorien



9.7 Wireless Markup Language – WML

WML steht für »Wireless Markup Language«, die Darstellungssprache von WAP (Wireless Application Protocol). Da man nicht nur auf statische WAP-Seiten angewiesen sein will, liegt eine Verbindung von PHP und WML nahe.

9.7.1 Grundlagen WML

In diesem Abschnitt finden Sie einige elementare Informationen und Hinweise auf weiterführende Seiten im Internet. Vor der praktischen Arbeit sollten Sie einige theoretischen Überlegungen zum Sinn des Projekts anstellen. Wichtig ist, dass bei einer WAP-Homepage die Dienstleistung im Vordergrund steht. Niemand wird auf Ihrer Seite mit dem Handy surfen, um sich Bilder anzusehen. Zeit ist in diesem Fall wirklich Geld, deshalb ist es ratsam, die WAP-Seite besonders effektiv zu gestalten. Sind alle Fragen geklärt, beginnen Sie mit den Vorbereitungen des Webserver und der Programmierungsumgebung.

WML-Grundlagen

Reine WML-Kurse und eine WML-Referenz finden Sie hier:

http://7110.nokia.de/wapkurs/wapkurs_set.html

http://www.wap-forum.in-X.de/grl_code.htm

Voraussetzungen im Webserver

Um erfolgreich mit WML arbeiten zu können, muss der Webserver die folgenden MIME-Typen beherrschen:

Beschreibung	MIME-Typ	Erweiterung
WML source	text/vnd.wap.wml	wml
Compiled WML	application/vnd.wap.wmlc	wmlc
WMLScript source	text/vnd.wap.wmlscript	wmls
Compiled WMLScript	application/vnd.wap.wmlscriptc	wmlsc
Wireless bitmap	image/vnd.wap.wbmp	wbmp

Tabelle 9.2:
MIME-Typen für
WML

Konsultieren Sie die Dokumentation des Webserver, um die Einstellungen der MIME-Typen vornehmen zu können.

Probleme mit den XML-Tags

Die Kurzform der PHP-Tags (`<? ... ?>`) ist *nicht* zu verwenden, da diese Form nicht dem XML-Standard entspricht. WML basiert aber auf XML, und deshalb muss es sich um ein wohlgeformtes XML-Dokument handeln. Umgekehrt scheitert PHP am XML-Deklarationstag.



Das folgende Beispiel zeigt, wie Sie mit PHP eine WML-Seite erzeugen und korrekt an das WAP-Handy senden. Das Problem stellt dabei das `<?xml`-Tag dar. PHP will den Code ab `<?` ausführen, was zu einer Fehlermeldung führt, weil das kein gültiger PHP-Code ist.

Eine mögliche Abhilfe: Statt `<?xml version="1.0"?>` außerhalb des PHP-Codes diesen innerhalb des PHP-Codes einfügen:

```
<?php
header("Content-Type:text/vnd.wap.wml\n\n");
echo "<?xml version=\"1.0\"?>";
?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
```

Listing 9.42:
Kopf einer WML-
Datei mit PHP
erzeugen

Die andere, regelkonformere Lösung besteht darin, die »Short-Tags« auszuschalten, also die Schreibweise `<? ... ?>` nicht mehr zuzulassen.

9.7.2 Praxis: WAP-Telefondatenbank

©-Hinweis

Das folgende Beispiel stammte ursprünglich von Thomas Fromm (tfromm@cs.uni-potsdam.de) und wurde unter der folgenden Adresse veröffentlicht:

```
http://fara.cs.uni-potsdam.de/index.html?thema=wap&titel=WAP
```

Es handelt sich um eine Datenbank mit Telefonnummern. Dieses Beispiel ist zwar ein relativ einfacher Dienst, Sie können sich damit aber einen guten Einblick verschaffen. Die Skripte wurden überarbeitet und weiterentwickelt. Auf der Website zum Buch finden Sie auch Links zu Emulatoren, um das Resultat auch ohne WAP-Handy sehen zu können.

Telefonauskunft per WAP-Handy

Die Datenbank

Bevor Sie mit der Programmierung beginnen, legen Sie sich eine Tabelle in Ihrer MySQL-Datenbank nach folgendem Schema an:

Listing 9.43:
Datenbanktabelle für
die Muster-
applikation
([telefonbuch.sql](#))

```
CREATE TABLE telefonbuch (  
    id INT(11) PRIMARY KEY AUTO_INCREMENT,  
    name VARCHAR(255),  
    vorname VARCHAR(255),  
    adresse VARCHAR(255),  
    wichtig TINYINT(1) DEFAULT '0' NOT NULL,  
    nummer VARCHAR(16)  
);
```

Aufbau einer WML-Seite

Eine WML-Seite besteht aus einem so genannten *Deck*, das in mehrere *Cards* unterteilt sein kann. Diese Karten repräsentieren die einzelnen Teilseiten der Homepage. Im Beispiel werden einige Cards benutzt, um bestimmte Informationen beim ersten Aufruf zu übermitteln; dazu gehören beispielsweise die wichtigsten Telefonnummern.

Die WML-Seite

Die Skripte werden mit *.PHP4 benannt. Es ist *nicht* zwingend erforderlich, dass jede WML-Datei auch *.WML heisst. Insbesondere, wenn sie dynamisch generiert wird, ist es eigentlich egal, sie muss nur den WML-Code an den WML-Browser ausgeben. Mit PHP muss die PHP-Engine den PHP-Code interpretieren. Das geht aber nur, wenn die Dateierweiterung PHP, PHP3 oder PHP4 lautet – je nach Version.

Jetzt muss die WML-Seite mit PHP ausgegeben werden. Das folgende Skript gibt nur statischen WML-Text aus, nur der Header wird mit PHP erzeugt. Machen Sie erst weiter, wenn dieses Skript im Handy oder Emulator funktioniert.

Auf den folgenden Seiten finden Sie in der Marginalspalte die Ausgabe für das Nokia 7110, erstellt mit dem Gelon-Emulator, den Sie auch über die Website zum Buch und <http://www.gelon.net> erreichen können.

```

<?php
// In dieser Datei werden die Datenbankdaten abgelegt
include("open.inc.php4");
header("Content-Type: text/vnd.wap.wml\n\n");
// Der XML-Kopf muss mit PHP erzeugt werden (wegen des <?-Tags)
echo "<?xml version=\"1.0\"?>";
?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<!-- die Startseite -->
<card id="Telefonbuch" title="Telefonbuch">
  <p>Bitte w&auml;hlen:<br/>
  <a href="#wichtig">Wichtige Nummern</a><br/>
  <a href="#listenamen">Liste sortiert nach Namen</a><br/>
</p>
</card>
<!-- Ende der Startseite -->
<!-- die Liste mit den wichtigsten Nummern -->
<card id="wichtig" title="Wichtige Nummern">
<p>Bitte w&auml;hlen:<br/>
<?php
// hier zwischen kommt denn die Liste mit den Nummern/Namen
// der Personen, die im Feld wichtig eine 1 haben
?>
</p>
</card>
<!-- Ende der Card mit den wichtigen Nummern -->
<!-- die Liste mit allen Einträgen sortiert nach Namen -->
<card id="listenamen" title="Liste sortiert nach Namen">
<p>Bitte w&auml;hlen:<br/>
<?php
// hier kommt die Liste sortiert nach Namen
// es ist eigentlich ratsam bei einer größeren Liste
// diese nicht sofort zu laden, sondern erst auf expliziten
// Wunsch z.B. durch ein Link auf ein separates PHP Skript
?>
</p>
</card>
</wml>

```

Listing 9.44:
Startseite der
Telefonbuch-
Applikation: [index](#)
(hier noch ohne
Funktion, nur zum
Testen der Reaktion
des Handys).
Ausgabe:



Die beiden notwendigen SQL-Anfragen müssten dem entsprechend den folgenden Mustern aussehen. Zuerst für die Card mit den wichtigen Nummern:

```
SELECT name,nummer FROM telefonbuch WHERE wichtig=1
```

Nun die Abfrage für die ganze Liste sortiert nach Namen:

```
SELECT name, nummer FROM telefonbuch ORDER BY name
```

Jetzt beginnt die Gestaltung des Decks mit den wichtigen Nummern (Erweiterung des Skripts in Listing 9.43):

Listing 9.45:
Anzeige der
Nummern
(Ausschnitt aus
[index1](#)). Ausgabe:



```
<!-- die Liste mit den wichtigsten Nummern -->
<card id="wichtig" title="Wichtige Nummern">
<p>Bitte wählen:<br/>
<?php
// Ausführen der Anfrage
$query = "SELECT name,nummer FROM telefonbuch WHERE wichtig=1"
$ergebnis=mysql_query($query);
mysql_close();
$eintraege=mysql_num_rows($ergebnis); // zählt die Einträge
// die for-Schleife zur Ausgabe der Einträge,
// muss bei 0 beginnen, da die mit mysql_result
// abgefragten Ergebnisse auch ab 0 gezählt werden
for ($x=0;$x<=$eintraege-1;$x++){
    // herauslesen des namens
    $name=mysql_result($ergebnis,$x,"name");
    // name braucht nur max. 15 Zeichen lang sein
    $name=substr($name,0,15);
    // Ausgabe des Namens und der Nummer
    echo"$name - ";
    echo mysql_result($ergebnis,$x,"nummer");
    // und noch ein Zeilenumbruch
    echo "<br/>";
}
?>
</p>
</card>
<!-- Ende der Card mit den wichtigen Nummern -->
```

Wenn Sie das nun in den Ausgangs Quellcode übertragen, so müsste das beim Anwählen der wichtigsten Nummern funktionieren.

Jetzt kommt die Liste sortiert nach Namen. Es ist ratsam, bei einer grösseren Liste diese nicht sofort zu laden, sondern erst auf expliziten Wunsch durch einen Link auf ein separates PHP-Skript die Verbindung zur Datenbank herzustellen.

Listing 9.46:
Ausgabe der
Nummernliste
(zweiter Teil des
Skripts in der Datei
[index1](#))

```
$ergebnis=mysql_query("SELECT name,nummer
                        FROM telefonbuch ORDER BY name;");
mysql_close();
$eintraege=mysql_num_rows($ergebnis);
```

Die for-Schleife zur Ausgabe der Einträge muss bei 0 beginnen, da die mit mysql_result abgefragten Ergebnisse auch ab 0 gezählt werden:

```
for ($x=0;$x<=$eintraege-1;$x++){
    $name=mysql_result($ergebnis,$x,"name");
    // Name auf 15 Zeichen kürzen wegen der Mini-Displays
    $name=substr($name,0,15);sein
    echo"$name<br/>";
    echo mysql_result($ergebnis,$x,"nummer");
    echo "<br/>";
}
?>
```

```

</p>
</card>
<!-- Ende der Card mit den sortierten Einträgen -->

```

Eine Nummer direkt anwählen

Es ist möglich, eine Nummer direkt anzuwählen. Allerdings erfolgt dies gerätespezifisch über das WTAI (Wireless Telephony Application Interface), welches nicht von allen Mobiltelefonen unterstützt wird.

Lassen Sie dazu die separate Angabe der Telefonnummer weg und Sie können direkt die Nummer über einen Link des Namens anwählen:

```

$nummer=mysql_result($ergebnis,$x,"nummer");
echo"<go href=\"wtai://wp/mc; $nummer\"/>$name</go>";

```

Des Weiteren haben manche Mobiltelefone – wie das Nokia 7110 – die Möglichkeit, über den »OPTIONS«-Schalter und die Option »Use number« alle Telefonnummern im Deck anzeigen zu lassen und auszuwählen.

Suchfunktionen

Eine weitere Funktion ist die Stichwortsuche. Dazu muss eine Zeichenkette an ein PHP-Skript übergeben werden. In WML wird das mit einer weiteren Card realisiert (Fortsetzung von Listing 9.46):

```

<!-- Beginn Suche -->
<card id="suche">
  <input format="text" name="suchtext" title="" value=""/>
  <anchor>Name suchen
    <go href="search.php4?suchtext=$(suchtext)"/>
  </anchor>
</card>
<!-- Ende Suche -->

```

Der Inhalt des Textfeldes *suchtext* wird in der URL an die MAIN.PHP übergeben. Das funktioniert auch bei Kontrollkästchen und anderen in WML definierten Formularelementen.

Außerdem wird das Auswahlménü um einen Eintrag erweitert, der die Anwahl des Decks erlaubt:

```

<a href="#suche">Nummern suchen</a><br/>

```

Im Skript SEARCH.PHP4 steht ganz zu Beginn eine Abfrage, ob die Variable *\$suchtext* initialisiert wurde. Dann wird eine Datenbankabfrage erstellt und je nach Ergebnis die Rückgabe gestaltet:

WTAI

*Listing 9.47:
Erweiterung um
Wählfunktionen:
[index2](#)*

*Listing 9.48:
Erweiterung um das
Suchformular
([index3](#)). Ausgabe:*



Listing 9.49:
Auswertung der
Suchanfrage und
Ausgabe der
Ergebnisse (search).
Ausgabe:



```
<?php
include("open.inc.php4");
if (isset($suchtext)) {
    $query = "SELECT nummer, name, id FROM telefonbuch WHERE ";
    $query .= "name LIKE '%$suchtext%' OR name LIKE '%$suchtext%'";
    $query .= "OR nummer LIKE '%$suchtext%' OR vorname LIKE ";
    $query .= "'%$suchtext%' OR adresse LIKE '%$suchtext%'";
    $ergebnis=mysql_query($query);
    mysql_close();
    $eintraege=mysql_num_rows($ergebnis);
    // WML erstellen
    header("Content-Type: text/vnd.wap.wml\n\n");
    echo "<?xml version='1.0'?">";
    echo "<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\"";
    echo "\"http://www.wapforum.org/DTD/wml_1.1.xml\">";
    echo "<wml>";
} else {
    $eintraege == 0;
}
if ($eintraege > 0){
    for ($x=0;$x<=$eintraege-1;$x++){
        echo "<card id=\"start\" title=\"Ergebnisse\">";
        echo "<p>";
        $name=mysql_result($ergebnis,$x,"name");
        $name=substr($name,0,15);
        $nummer=mysql_result($ergebnis,$x,"nummer");
        echo "$name<br/>";
        echo "<go href=\"wtai://wp/mc; $nummer\"/>$nummer<br/>";
    }
    echo "<a href=\"index3.php4\">Zur&uuml;ck</a><br/>";
    echo "</p></card>";
} else {
    echo "<card id=\"start\" title=\"\"> ";
    echo "<p>Keine Eintr&uuml;ge vorhanden.<br/> ";
    echo "<a href=\"index3.php4\">Zur&uuml;ck</a><br/></p></card>";
}
echo "</wml>";
?>
```

Einträge per Handy editieren

WAP dient vor allem besserer Interaktivität. Leider lassen viele WAP-Anwendungen das vermissen, obwohl WML hier genauso viel bietet wie HTML – die richtige Unterstützung auf der Serverseite vorausgesetzt. Mit einem weiteren Skript wird dies realisiert. Im Gegensatz zur ersten Version wird nur der folgende Link im Auswahlmenü hinzugefügt:

```
<anchor>Name zu bearbeiten:
  <go href="edit.php4?edit=form"/>
</anchor>
```

Listing 9.50:
Zum Editieren
startet ein weiteres
Skript ([index4](#))

Wie Sie sehen, zeigt der Link zum Editieren auf ein separates Skript. Dieses Skript EDIT.PHP wird nachfolgend gezeigt. Es realisiert vor allem Eingabefelder. Bedenken Sie bei der Ausarbeitung von Formularen, dass die Bedienung mit einem Handy ausgesprochen unkomfortabel ist. Je nach Modell wird dabei teilweise in einen eigenen Modus gesprungen, wo mit der mehrfach belegten Tastatur die Daten erfasst werden. Umfangreiche Formulare sind wenig hilfreich. Alternativ können Links verwendet werden, die eine Vorauswahl geben, kombiniert mit einer einfachen Suchfunktion.



Zuerst der WML-Teil, in dem ein Formular zum Erfassen eines Eintrags erzeugt wird:

```
<?php
if ($edit == "form"):
?>
<wml>
  <card id="start" title="Edit Eintrag">
    Name:<br/>
    <input format="text" name="name" value=""/>
    Vorname:<br/>
    <input format="text" name="vorname" value=""/>
    Adresse:<br/>
    <input format="text" name="adresse" value=""/>
    Nummer:<br/>
    <input format="*N" name="nummer" value=""/>
    <anchor>[Normal]
    <go href="edit.php?wichtig=0&nummer=$(nummer)&
      name=$(name)&vorname=$(vorname)&edit=0K"/>
    </anchor>
    <anchor> [Wichtig]
    <go href="edit.php?wichtig=1&nummer=$(nummer)&
      name=$(name)&vorname=$(vorname)&edit=0K"/>
    </anchor>
    <br/>
    <a href="index4.php4">Abbrechen</a><br/>
  </card>
</wml>
<?php
endif;
?>
```

Listing 9.51:
Der Formular-Teil
des Skripts [edit](#).
Ausgabe:



Dann folgt der PHP-Teil (im selben Skript), der die erfassten Daten in die Datenbank einträgt. Sie können sich vom Erfolg überzeugen, indem Sie eines der Skripte starten, die die Liste der Namen anzeigen oder die Suche erlauben.

Listing 9.52:
Bestätigung der
Eintragung einer
neuen Adresse (*edit*).
Ausgabe:



```
<?php
if ($edit == "OK") :
    $query = "INSERT INTO adresse (name, vorname, adresse,
                                   nummer, wichtig) ";
    $query .= "VALUES ('$name', '$vorname', '$adresse',
                       '$nummer', '$wichtig')";
    $result = mysql_query($query);
    ?>
<wml>
    <card id=\"eingetragen\" title=\"Adressbucheintrag\">
        <p>Adressbucheintrag aufgenommen:</p>
        <p>Name: " <?php echo $name ?>"</p>
        <p>Vorname : "<?php echo $vorname ?>"</p>
        <p>Adresse : "<?php $adresse ?>"</p>
        <p>Telefon : "<?php $telefon ?>"</p>
        <p>
            <anchor>Zurück
                <go href="index4.php4"/>
            </anchor>
        </p>
    </card>
</wml>
<?php
endif;
?>
```

Nach dem Eintrag mit INSERT, der hier ohne weitere Fehlerprüfung erfolgt, werden die empfangenen Daten noch einmal ausgegeben. Den Eintrag des Feldes *wichtig* steuert der Link, der das Formular absendet. Dies ist ein kleiner Trick in Ermangelung von Checkbox-Elementen in WML.

9.7.3 Praktischer Einsatz von WAP/WML

Diese Beispiele sollen nur als Anregung für die weitere Beschäftigung mit WAP/WML dienen und zeigen, wie man anderes als HTML erzeugt. Ebenso wie bei HTML gehören dazu aber gute WML-Kenntnisse. Hier sollten Sie sich vor den ersten Schritten unbedingt mit der passenden Literatur versorgen, die es inzwischen reichlich gibt.

Fehlersuche

Die Fehlersuche ist mit WML-Dateien nicht einfach. Der Browser kann HTML-Seiten nicht darstellen. Die WML-Emulatoren weisen in der Regel nur mit einer allgemeinen Meldung darauf hin, dass keine WML-Daten erkennbar waren. HTML wird aber dennoch erzeugt, wenn der PHP-Parser Fehler erkennt und seine normale Fehlerseite ausgibt. Versuchen Sie dann, die Seite mit einem normalen Browser

anzuzeigen. Der Browser wird die WML-Tags ignorieren oder, wenn Sie denen in HTML ähneln, problemlos anzeigen. Wenn keine Parserfehler mehr auftreten, dann senden Sie die Seite an das Handy.

Etwas problematischer ist das Erkennen von Laufzeitfehlern. Hier erzeugt die PHP-Engine Fehler erst nach dem korrekten Aufbau der WML-Seite. Der Browser im Handy kann aber die Ausgaben nicht anzeigen, häufig bleiben Fehlermeldungen deshalb unbemerkt. Hier hilft nur saubere Programmierung, das heißt, alle Fehlerzustände, die theoretisch auftreten können, müssen explizit behandelt werden und zu klar erkennbaren Fehlerausgaben führen. Sinnvoll sind kurze Fehlernummern, die Sie sich vorher überlegt haben. Lange Meldungstexte haben sicher auf einem Handy-Display nichts zu suchen.

Hilfe im Internet

Wenn Sie sich intensiver mit WAP/WML beschäftigen müssen, können Ihnen die folgenden Adressen helfen:

<http://allnetdevices.com/faq>, FAQ rund um WML/WAP

<http://www.wapmagazin.de>, WAP-Magazin

<http://de.mobile.yahoo.com>, Yahoo! Mobil

<http://www.nokia.com/corporate/wap/index.html>, Nokia WAP-Seiten

9.8 Web Distributed Data Exchange – WDDX

WDDX steht für »Web Distributed Data Exchange«. Darunter wird eine XML-basierte (siehe dazu ➡ Abschnitt 9.6 *Extensible Markup Language – XML* ab Seite 697) Technologie verstanden, die den Austausch komplexer Daten zwischen Programmiersprachen erlaubt.

9.8.1 WDDX im Detail

Zu WDDX existiert eine XML 1.0 konforme DTD. Die Übertragung der Daten erfolgt über die bekannten Internetprotokolle HTTP, SMTP, POP, FTP und andere, die zum Transport von Textdateien eingesetzt werden können.



WDDX wurde von der Firma Allaire entwickelt, um Probleme beim Datenaustausch zwischen Webanwendungen zu lösen. Allaire ist der Entwickler des bekannten Webapplikations-Servers ColdFusion. Simeon Simeonov, der Architekt von ColdFusion, entwickelte WDDX speziell zum Datenaustausch mit ColdFusion. Die Arbeit wurde universeller und allgemeingültiger und so entstand die WDDX.org. Nate Weiss entwickelte die Idee weiter zu einer unabhängigen Program-

Historie

mierplattform (WDDX SDK) und so ist WDDX inzwischen auch in anderen Applikationssprachen verfügbar.

WDDX ist derzeit noch kein formeller W3C-Standard. WDDX ist aber eine freie Software und hat deshalb bereits eine gute Verbreitung gefunden. Die Basis XML 1.0, auf der WDDX beruht, ist allerdings ein offizieller W3C-Standard.

Wie funktioniert WDDX?



Jede Programmier- und Skriptsprache, wie PHP, Perl, ASP, Java, JavaScript oder auch ColdFusion besitzt interne native Datenstrukturen zum Ablegen von Daten, dargestellt als Arrays, Variable, Datensätze oder Objekte. WDDX liefert für jede Sprache ein Modul, das diese internen Daten serialisiert und in einer abstrakten Form repräsentiert. Diese Form wird durch XML beschrieben. Sie können beispielsweise mit Hilfe von WDDX ein komplexes Array in ColdFusion serialisieren, nach XML konvertieren, an einen ASP-Server senden und dort aus XML dekodieren, deserialisieren und als VBScript-Array wieder zur Verfügung stellen. Durch XML wird dieser Prozess für Entwickler transparent.

Web Syndication

Web Syndication ist eine Idee, wie Daten und Dienste einer Website anderen Sites zur Verfügung gestellt werden können. Netzwerke aus solchen Sites können damit komplexe Dienste an vielen Stellen zur Verfügung stellen. Praktische Beispiele sind das Amazon-Partnerprogramm oder Zahlungssysteme wie CyberCash. Dies sind zwar (noch) keine WDDX-Anwendungen, zeigen aber, welche Arten von Diensten bereits heute für andere Nutzer bereitgestellt werden können.

Mit WDDX wird die Verteilung solcher Dienste extrem einfach, denn die Übertragung von Daten kann nun auch direkt zwischen Systemen erfolgen. Wenn der Server des Anbieters mit Perl arbeitet und der Client mit PHP, kann er direkt die Daten abrufen und diese im Skript, in nativen Datenstrukturen, bereitstellen. WDDX hat keine speziellen Funktionen für Web Syndication, sondern bietet eine generelle Entwicklungsumgebung für derartige Anwendungen. Funktionen auf Anwendungsebenen (beispielsweise Nutzerautorisierung oder Zeitplaner) sind nicht vorhanden. Die Programmiersprachen, wie PHP, in die WDDX eingebettet ist, bieten für das Applikationsniveau bereits eine exzellente Unterstützung.

WDDX ist allerdings nicht nur auf die Anwendung bei Web Syndication beschränkt. Es gibt viele sehr nützliche Anwendungen, die im Folgenden noch beschrieben werden. Es geht um generellen Datenaustausch, zwischen Browser und Server, für lokale Anwendungen (offline) oder Server-zu-Server-Verbindungen.

Grundsätzlich können mit WDDX Daten in einem universellen Format im Netzwerk bereitgestellt werden.



Mit WDDX ist es möglich, die Informationskette der Kundenbeziehungen in elektronischer Form abzubilden. So kann ein Großhändler die Daten seiner Produkte per WDDX seinen Händlern bereitstellen, die ihrerseits die aktuellsten Daten in elektronischen Shops ihren Kunden anbieten. Shopassistenten und Suchmaschinen können wiederum mit ihren speziellen Anforderungen auf die Daten zurückgreifen und in ihre Angebote einbauen. Das ändert nichts daran, dass die eigentliche kommerzielle Transaktion auf dem Server desjenigen abläuft, der die Daten ursprünglich bereitstellte. Die Idee ist revolutionär, denn es bleibt trotz der technisch differenzierten Verbindung eine klassische Vertriebsstruktur erhalten. Der Kunde profitiert von den Leistungen der Anbieter, die Beratung, Erfahrung, Spezialisierung und direktere Unterstützung bieten.

Vorteile

Webentwickler können einige kritische Probleme mit WDDX lösen. Für jeden Fall gibt es spezielle Sprachen, die sich ideal zur Lösung eines Applikationsproblems eignen. Daten sind aber Daten, eine gemeinsame Datenplattform wäre der Idealzustand. Dem kommt WDDX ziemlich nahe, zumal teure, technisch aufwändige und komplexe Datenbanksysteme nicht notwendig sind. Es lassen sich Applikationen entwickeln, die nicht mehr von einer bestimmten Sprache abhängig sind – eine signifikante Steigerung der Plattformunabhängigkeit.

Lösungen

Dabei ist WDDX nicht auf den Server beschränkt. Clientseitige Applikationen aus JavaScript oder VBScript lassen sich ebenso einbinden. Durch die Verlagerung der Intelligenz in die ohnehin starken Clients (Thin-Clients sind ja Ende der 90er Jahre fast wieder tot) lassen sich die kostbaren Serververbindungen spürbar entlasten.

Jede Applikation, die Daten mit einer anderen Applikation über das Web austauschen möchte, kann von WDDX profitieren. Das betrifft Business-to-Business-Anwendungen, Extranet-Applikationen und andere Gelegenheiten, bei denen Unternehmen Daten über die Wertschöpfungskette austauschen. Dabei spielt es keine Rolle, ob es sich um Kundendaten, Produktinformationen oder betriebsinterne Daten handelt. Da WDDX sprachunabhängig ist und problemlos auf HTTP aufsetzt, kann jedes Unternehmen mit der ihm genehmen Software arbeiten; sei es ASP, ColdFusion, Perl oder eben auch PHP. Die Verwendung der bereits etablierten Plattform vereinfacht die Anwendungsentwicklung erheblich und vermeidet die beim Zusammenschalten von Applikationen unterschiedlicher Sprachen oft auftauchenden Probleme. Auch als Brücke zwischen dem klassischen Windows-Desktop und dem Web kann WDDX eingesetzt werden. Oft benötigen Unternehmen Programme, die mit Visual Basic, C++, Del-

Wofür ist WDDX geeignet?

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

phi oder anderen Entwicklungsumgebungen für Windows erstellt wurden und mit Webservern Daten austauschen müssen. Mit WDDX haben Entwickler die Möglichkeit, ihre Applikationen an HTTP-Server zu binden. Serverseitig muss dabei kein neuer Code entwickelt werden. Die Nutzung von WDDX beispielsweise über COM erlaubt dabei den einfachen und sauberen Datenaustausch, ohne dass man gezwungen wäre, auf einer der beiden Seiten nochmals zusätzliche Technologien einzuführen.

Das WDDX SDK

Das WDDX SDK²⁹ ist eine Entwicklungsumgebung, die es erlaubt, verteilte Webapplikationen mit WDDX zu entwickeln. Das SDK selbst ist kostenlos von der WDDX-Website <http://www.wddx.org> zu beziehen.

Voraussetzungen

Um WDDX einsetzen zu können, wird eine Plattform vorausgesetzt, die XML unterstützt. Das SDK selbst ist als ZIP- oder TAR-Datei erhältlich. Als Grundlage dient ein Webbrowser, der HTML darstellen kann. Viele Beispiele im SDK nutzen ASP, Perl, ColdFusion oder Java, die sich teilweise leicht nach PHP übertragen lassen. Trotz des Aufsatzes auf XML werden vom Entwickler keine oder nur sehr oberflächliche XML-Kenntnisse verlangt.

Unterstützte Datentypen

WDDX unterstützt die folgenden Datentypen:

- Boolean (TRUE/FALSE, Wahr/Falsch)
- Number (Zahlen)
- Date-Time (Datum und Uhrzeit) nach ISO 8601
- String (Zeichenkette)

Die Zeitangabe nach ISO 8601 enthält Zeitzoneinformationen, die von WDDX erkannt und entsprechend den lokalen Bedingungen umgesetzt wird. Auf der Basis der Grunddatentypen werden auch komplexe Strukturen behandelt:

- Arrays (Felder)
- Structures (Strukturen, damit sind hier assoziative Felder gemeint)
- Datensätze (Anordnung in starren Tabellen)

Browser-unterstützung

Die Daten werden in WDDX für den Browser entsprechend aufbereitet. Neben HTML wird zur Ausgabe JavaScript 1.1 verwendet. Alle Browser ab Version 3 sollten diese Informationen darstellen können.

²⁹ SDK = Software Development Kit, Satz von Software-Entwicklungswerkzeugen

Da WDDX bei der Ausgabe von Daten auf HTTP aufsetzt, können die bekannten Sicherheitsmaßnahmen mit SSL uneingeschränkt genutzt werden. WDDX-Pakete sind immer Zeichenketten, die problemlos verschlüsselt und entschlüsselt werden können.

Sicherheit

Standard-XML nutzt das Document Object Model (DOM), um auf XML-Dokumente zuzugreifen und diese zu manipulieren. In diesem Modell hat der Entwickler den vollen Zugriff auf alle Programmelemente. In WDDX muss der Programmierer nicht auf diese Elemente zugreifen, denn es existieren sprachspezifische Erweiterungen für fast alle Programmier- und Skriptsprachen, die den Zugriff stark vereinfachen.

XML und WDDX

9.8.2 WDDX-Elemente

Einführung

Die WDDX-DTD wird verwendet, um WDDX-Dokumente zu überprüfen und auszuwerten. Nachfolgend ein Beispiel für ein typisches WDDX-Dokument, eine Muster-DTD wird weiter unten gezeigt:

**WDDX Document
Type Definition
(DTD)**

```
<?xml version='1.0'?>
<!DOCTYPE wddxPacket SYSTEM 'wddx_0090.dtd'>
<wddxPacket version='1.0'>
<header/>
  <data>
    <struct>
      <var name='s'>
        <string>a string</string>
      </var>
      <var name='n'>
        <number>-12.456</number>
      </var>
      <var name='d'>
        <dateTime>1998-06-12T04:32:12</dateTime>
      </var>
      <var name='b'>
        <boolean value='true' />
      </var>
      <var name='a'>
        <array length='2'>
          <number>10</number>
          <string>second element</string>
        </array>
      </var>
      <var name='obj'>
        <struct>
          <var name='s'>
            <string>a string</string>
          </var>
```

*Listing 9.53:
Eine typische
WDDX-Datei*

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

```

        <var name='n'>
            <number>-12.456</number>
        </var>
    </struct>
</var>
<var name='r'>
    <recordset rowCount='2' fieldNames='NAME,AGE'>
        <field name='NAME'>
            <string>John Doe</string>
            <string>Jane Doe</string>
        </field>
        <field name='AGE'>
            <number>34</number>
            <number>31</number>
        </field>
    </recordset>
</var>
</struct>
</data>
</wddxPacket>

```

Hier wird ein Basisobjekt als assoziatives Array³⁰ mit sechs Eigenschaften definiert:

- *s* als Zeichenkette 'a string',
- *n* als Zahl -12.456,
- *d* als Datum mit dem Wert June 12, 1998 4:32:12am,
- *b* als Boolescher Wert TRUE,
- *a* als Array mit zwei Elementen (10 und 'second element'),
- *obj* als assoziatives Array mit zwei Eigenschaften *s* und *n* sowie *r* als Datensatz mit zwei Reihen und den Feldern *NAME* und *AGE*.

WDDX-Datentypen

WDDX kennt folgende elementare Datentypen:

- *Numbers*

Zahlen sind grundsätzlich Gleitkommazahlen mit einem Wertebereich von $\pm 1,7^{\pm 308}$. Die Genauigkeit ist auf 15 Stellen nach dem Komma begrenzt. Die interne Darstellung benötigt 8 Bytes.

- *Date-Time*

Datums- und Zeitwerte entsprechen der ISO 8601, haben also beispielsweise das Format 1964-26-5T09:05:32+2:0. Einstellige Anga-

Elementare Datentypen

³⁰ In der WDDX-Literatur ist dafür oft der Begriff Structur (Struktur) zu finden.

ben für Monat, Tag, Stunde oder Minute benötigen keine führende Null. Die Angabe der Zeitzone ist optional, wird jedoch für die Übersetzung in lokale Zeitangaben benötigt. Die Darstellform ist Jahr-2000-sicher, das Jahr muss immer vierstellig angegeben werden.

- *Strings*

Zeichenketten können eine unbegrenzte Länge haben und sollten keine Null-Zeichen enthalten. Um erweiterte Zeichen darstellen zu können, werden UTF-8-Hexadezimalcodes aus zwei Bytes eingesetzt.

Zeilenendezeichen sind plattform- und programmiersprachenabhängig. Oft wird ein einfacher Zeilenumbruch eingesetzt (x0A), ein Wagenrücklauf (x0D) oder eine Kombination aus beidem (x0D0A). Die XML-Spezifikation verlangt, dass alle Zeilenumbrüche in die Variante x0A konvertiert werden. Eben dies erfolgt auch in WDDX.

**Behandlung von
Zeilenumbrüchen**

WDDX kennt drei komplexe Datentypen:

**Komplexe
Datentypen**

- *Arrays*

Arrays werden durch numerische Indizes adressiert und können jeden elementaren Datentyp enthalten. Der Index beginnt mit der 0 (Null).

- *Structures*

Strukturen sind assoziative Arrays, deren Indizes durch Zeichenketten gebildet werden. Groß- und Kleinschreibung spielt keine Rolle.

- *Recordsets*

Datensätze bilden tabellarische Daten ab: ein Satz von benannten Feldern mit einer Anzahl Reihen. Wenn dieses starre Modell nicht ausreicht, sollten Arrays mit Elementen aus Strukturen gebildet werden, was weitaus flexibler ist. Datensätze eignen sich aber gut zur Übernahme von Daten aus SQL-Abfragen. Groß- und Kleinschreibung bei den Feldnamen spielt keine Rolle.

Die folgende Tabelle zeigt einen Vergleich der Datentypen der wichtigsten Sprachen:

WDDX Type	COM	JAVA	PHP	JavaScript
boolean	boolean	boolean	int	boolean
number	Float	float	float	number
dateTime	DATE	??	string	Date
string	BSTR	String	string	string

Tabelle 9.3:
WDDX-Datentypen
im Vergleich (?? =
der Typ wird nicht
direkt unterstützt)

WDDX Type	COM	JAVA	PHP	JavaScript
array	VARIANT array	??	array	Array
struct	IWDDXStruct	??	array	Object
recordset	IWDDXRecordset	??	??	WddxRecordset

Besonderheiten

- Null-Werte** Eine explizite Übertragung von Null-Werten erfolgt nicht. Bei der Serialisierung werden Null-Werte zu leeren Zeichenketten. Bei der Deserialisierung muss die entsprechende Komponente für die korrekte Behandlung sorgen.
- Serialisierungsmodell** WDDX nimmt nur unmittelbar vorliegende Daten an, keine Verknüpfungen oder Objektreferenzen. Aliase werden aufgelöst und lassen mehrere Objekte entstehen. Zyklische Referenzen können zu unvorhersehbaren Ergebnissen oder Endlosschleifen führen. Die konkrete Reaktion hängt von der Implementierung der Funktion ab.
- DTD Empfehlungen** Es ist empfehlenswert, die DTD wortreich und beschreibend aufzubauen. WDDX verwendet intern Kompressionsalgorithmen, um die Übertragungsbandbreite zu reduzieren. Es ist nicht sinnvoll, bei der Beschreibung der DTD kryptische Abkürzungen zu nutzen, die später die Wartung der Datei erschweren.

Muster-DTD

Nachfolgend finden Sie eine Muster-DTD für WDDX-Dateien der Version 0.9.

Listing 9.54:
Eine DTD für
WDDX

```
<!ELEMENT wddxPacket (header, data)>
<!ATTLIST wddxPacket
    version CDATA #FIXED "0.9">
<!ELEMENT header (comment?)>
<!ELEMENT comment (#PCDATA)>
<!ELEMENT data (boolean
    | number
    | dateTime
    | string
    | array
    | struct
    | recordset)*>
<!ELEMENT boolean EMPTY>
<!ATTLIST boolean
    value (true | false) #REQUIRED>
<!ELEMENT string (#PCDATA | char)*>
<!ELEMENT char EMPTY>
<!ATTLIST char
    code CDATA #REQUIRED>
<!ELEMENT number (#PCDATA)>
```

```

<!ELEMENT dateTime (#PCDATA)>
<!ELEMENT array (boolean
    | number
    | dateTime
    | string
    | array
    | struct
    | recordset)*>
<!ATTLIST array
    length CDATA #REQUIRED>
<!ELEMENT struct (var*)>
<!ELEMENT var (boolean
    | number
    | dateTime
    | string
    | array
    | struct
    | recordset)>
<!ATTLIST var
    name CDATA #REQUIRED>
<!ELEMENT recordset (field*)>
<!ATTLIST recordset
    rowCount CDATA #REQUIRED
    fieldNames CDATA #REQUIRED>
<!ELEMENT field
    (boolean | number | dateTime | string)*>
<!ATTLIST field
    name CDATA #REQUIRED>

```

9.8.3 WDDX-Funktionen in PHP

PHP unterstützt WDDX direkt, ohne den Umweg über XML. Damit lassen sich besonders einfach Cross-Plattform-Anwendungen schreiben.

Funktionsübersicht

Eine Beschreibung der WDDX-Funktionen mit kurzen Beispielen finden Sie in der Referenz. Hier eine kurze Funktionsübersicht:

- `wddx_serialize_value`
Serialisiert einen Wert in ein WDDX-Paket.
- `wddx_serialize_vars`
Serialisiert mehrere Werte.
- `wddx_packet_start`
Startet ein neues Paket.

- `wddx_packet_end`
Beendet ein Paket.
- `wddx_add_vars`
Beendet ein Paket mit einer spezifischen ID.
- `wddx_deserialize`
Deserialisiert ein Paket.

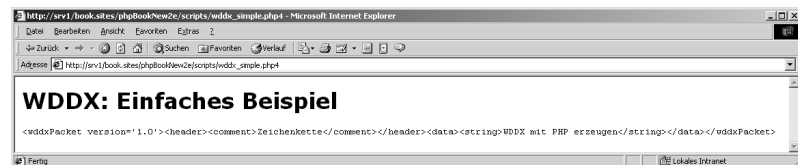
Eine WDDX-Beispielanwendung

Listing 9.55:
Einfachste WDDX-
Beispielanwendung:
`wddx_simple`

```
<?php
$data = "WDDX mit PHP erzeugen";
$comment = "Zeichenkette";
echo '<pre>';
$wddx = wddx_serialize_value($data, $comment);
echo htmlspecialchars($wddx);
echo '</pre>';
?>
```

Dieses Beispiel gibt Folgendes aus:

Abbildung 9.24:
Anzeige eines
WDDX-Paketes



Die Darstellung ist nicht geeignet, größere Datenmengen anzuzeigen. Deshalb wird der bereits vorgestellte XML-Parser in einer modifizierten Form verwendet, um die Ausgabe aufzubereiten. Das folgende Skript zeigt das Ergebnis.

Listing 9.56:
Komplexeres
WDDX-Beispiel:
`wddx_complex`

```
<style>
.tag {color:blue}
.attr {color:red}
.parm {color:green}
</style>
<?php
$depth = 0;
function startElement($parser, $name, $attrs) {
    global $depth;
    echo str_repeat("&nbsp;", $depth*4);
    echo "&lt;&lt;span class=tag>$name</span>";
    if (is_array($attrs)) {
        foreach($attrs as $x => $y) {
            echo " <span class=attr>$x</span>=";
            echo "'<span class=parm>$y</span>'";
        }
    }
}
```

```

        echo "&gt;<br>";
        $depth++;
    }

    function endElement($parser, $name) {
        global $depth;
        $depth--;
        echo str_repeat("&nbsp;", $depth*4);
        echo "<span class=tag>&lt;/>$name&gt;</span><br>";
    }

    function cdata($parser, $data)
    {
        global $depth;
        echo str_repeat("&nbsp;", $depth*4);
        echo "<b>";
        echo htmlspecialchars($data);
        echo "</b><br>";
    }

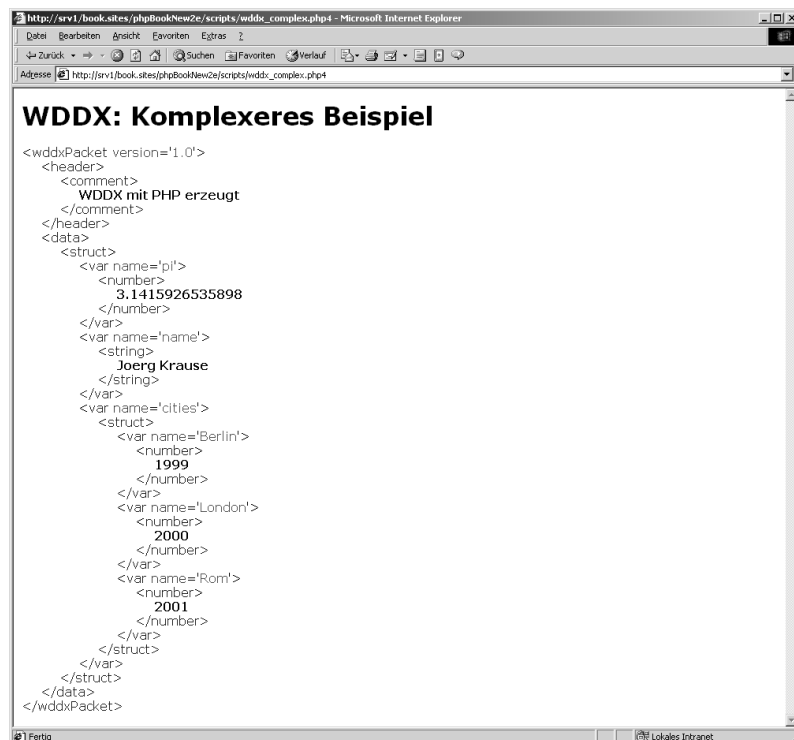
    $pi = M_PI;
    $name = "Joerg Krause";
    $cities = array("Berlin" => 1999,
                   "London" => 2000,
                   "Rom" => 2001);
    $packet_id = wddx_packet_start("WDDX mit PHP erzeugt");
    wddx_add_vars($packet_id, "pi");
    wddx_add_vars($packet_id, "name");
    wddx_add_vars($packet_id, "cities");
    $packet = wddx_packet_end($packet_id);
    $xml_parser = xml_parser_create();
    xml_set_element_handler($xml_parser, "startElement", "endElement");
    xml_set_character_data_handler($xml_parser, "cdata");
    xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, 0);
    if (!xml_parse($xml_parser, $packet)) {
        die(sprintf("XML-Fehler: %s auf Zeile %d",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
    xml_parser_free($xml_parser);
?>

```

Dieses Beispiel gibt folgendes Paket aus:

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Abbildung 9.25:
Ausgabe eines
WDDX-Pakets mit
XML-Funktionen



In der Praxis werden solche Pakete dann als Datei abgelegt und per FTP oder HTTP versendet oder in Datenbanken gespeichert. Interessant mag auch die Nutzung der XML-Funktionen zur Analyse der WDDX-Pakete sein.

Wie es funktioniert

Zuerst werden die Behandlungsroutinen für die Tags, Attribute und Daten definiert. Besonderheiten sind hier nicht zu finden – die Einrückung wird mit der `str_repeat`-Funktion vorgenommen. Abgesehen von der im Beispiel verwendeten Färbung ist der Fantasie hier keine Grenze gesetzt, was die weitere Verarbeitung betrifft.

Dann folgt die Definition einiger Musterwerte. Nach der Erzeugung eines neuen WDDX-Blocks im Speicher werden diese Variablen dem Paket zugewiesen:

```
$packet_id = wddx_packet_start("WDDX mit PHP erzeugt");
wddx_add_vars($packet_id, "pi");
wddx_add_vars($packet_id, "name");
wddx_add_vars($packet_id, "cities");
```

Das fertige Paket wird in einer Variablen gespeichert:

```
$packet = wddx_packet_end($packet_id);
```

Dann wird ein XML-Parser erzeugt, der sich um die Darstellung des WDDX-Paketes kümmert:

```
$xml_parser = xml_parser_create();
```

Drei Behandlungsfunktionen werden benötigt. Zwei für die öffnenden und schließenden Tags und eine für die enthaltenen Daten:

```
xml_set_element_handler($xml_parser, "startElement", "endElement");  
xml_set_character_data_handler($xml_parser, "cdata");
```

Standardmäßig werden alle Tag-Namen in Großbuchstaben verwandelt, was bei der Anzeige manchmal störend ist.

```
xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, 0);
```

Dann erfolgt die Verarbeitung in einem Zug, anschließend wird der Parser wieder freigegeben. Die Fehlerausgabe sollte bei der automatischen Erzeugung von WDDX-Paketen nicht reagieren:

```
if (!xml_parse($xml_parser, $packet)) {  
    die(sprintf("XML-Fehler: %s auf Zeile %d",  
        xml_error_string(xml_get_error_code($xml_parser)),  
        xml_get_current_line_number($xml_parser)));  
}  
xml_parser_free($xml_parser);
```

V

S

I

1

2

3

4

5

6

7

8

9

10

11

A

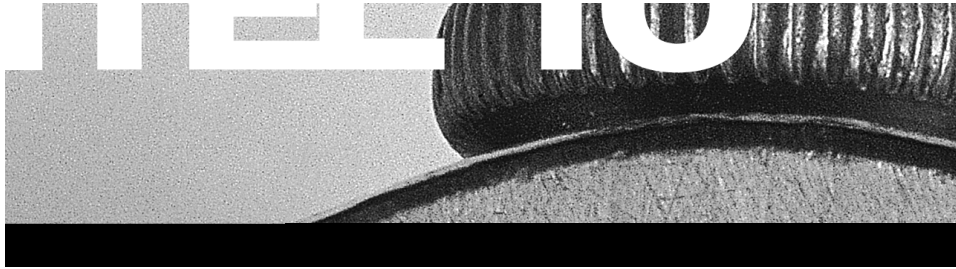
B

C

D

E

10



Zusatzprogramme

In diesem Kapitel werden Editoren und die von Zend entwickelten Zusatzprogramme vorgestellt. Einige dieser Produkte sind kostenpflichtig, zum Ausprobieren steht jedoch meist eine Testversion zur Verfügung. Ausführlicher wird der einzige derzeit verfügbare Debugger, der Bestandteil der ZendIDE ist, vorgestellt.

Editoren (Seite 749)

ZendIDE (Seite 750)

Zend-Hilfsprogramme (Seite 762)

10.1 Editoren

Gute Werkzeuge sind schon immer die Basis für solides Handwerk gewesen. Daran ändert sich in der Softwarewelt nichts. Wer professionell programmiert, sollte auch professionelles Werkzeug verwenden.

10.1.1 Editoren für Windows

Die folgenden Editoren stehen zwar erst am Anfang der Entwicklung, fallen aber durch die direkte Unterstützung von PHP auf. Sie sind für Windows-Plattformen ausgelegt.

Es gibt eine ganze Reihe Editoren, die für PHP geeignet sind, d.h. bei denen die Syntaxprüfung und farbliche Darstellung durch eine externe Datei einstellbar oder fest integriert ist. In solchen Fällen kann man den PHP-Befehlsvorrat mit überschaubarem Aufwand definieren und einen PHP-tauglichen Editor schaffen.

Übersicht

Die folgende Liste zeigt eine Auswahl – ohne Anspruch auf Vollständigkeit:

- PHPEd

<http://www.soysal.com/PHPEd>

- UltraEdit-32

<http://www.idmcomp.com>

- Homesite 4.5

<http://www.allaire.com>

Dies ist ein kommerzielles Produkt, das in Deutschland ca. 300 DM kostet. Für Hobbyentwickler sicher etwas zu teuer, für Profis ist der Editor aber durchaus empfehlenswert, auch für HTML.

- EditPlus

<http://www.editplus.com>

- Webauthor

<http://www.resoft.de>

- WeaverSlave

<http://www.weaverslave.de>

Ein kleiner, sehr schneller Editor mit Basisfunktionen. Für einfache Ansprüche, aber mit Syntaxhervorhebung.

V

S

I

1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E

10.1.2 Editoren für Unix

Auch unter Unix gibt es viele Editoren. Da hier eher der Einsatz auch außerhalb der Programmierung erforderlich ist, haben sich die meisten Nutzer schon für einen entschieden. Eine ausführliche Vorstellung dürfte hier deshalb weniger auf Interesse stoßen. Eine kurze Übersicht soll Einsteigern die Möglichkeiten zeigen.

Übersicht

Editoren für Unix sind jedem Unix/Linux-Benutzer sicher bekannt. Empfehlenswert sind natürlich Programme, die auf die Syntax von PHP reagieren und eine entsprechende farbliche Unterstützung bei der Anzeige bieten. Die folgenden Adressen sind eine Anlaufstelle:

- CForce

<http://www.codeforce.com>

Der Editor CForce ist ein professionelles Produkt, das sehr viele Programmiersprachen unterstützt. Mit \$150 ist eine einzelne Lizenz aber relativ teuer.

- NEdit

<http://www.nedit.org>

Ein kleiner und einfacher Freeware-Editor, der PHP seit kurzem unterstützt. Der grafische Editor ist für X-Window Systeme geeignet.

- VIM

<http://www.vim.org>

Der bekannte VI in der Version VI Improved (= VIM) wird für nahezu alle Plattformen angeboten. VIM ist sehr schnell, reich an Funktionen und wird im Internet gut unterstützt.

- XEMacs

<http://www.xemacs.org>

Der noch bekanntere XEmacs in der neuesten Version beherrscht unter anderem auch SGML und XML.

10.2 ZendIDE

Die ZendIDE ist eine vollständige Entwicklungsumgebung, basierend auf Java. Sie läuft sowohl unter Windows als auch unter Unix. Voraussetzung ist Java Runtime Environment Version 1.3. Das Programm

kann für \$ 250 bei Zend erworben werden, eine 30-Tage-Testversion ist verfügbar.

10.2.1 Vorbereitung

Da die ZendIDE in Java entwickelt wurde, muss zuerst Java installiert werden. Sie erhalten die benötigte Version von Sun unter folgender Adresse:

Laden der Java-Laufzeitumgebung

```
http://java.sun.com/j2se/1.3/jre
```

Wählen Sie dort Ihr Betriebssystem (Solaris, Linux oder Windows) aus und bestätigen die Lizenzvereinbarung. Die Größe der Datei beträgt etwa 8 Megabyte.

Nun entpacken Sie die beiden Pakete der ZendIDE – eines für Client (der Editor und Debugger) und eines für den Server.

Entpacken der ZendIDE-Dateien

Einschränkungen in der Benutzung

Es spielt eigentlich keine Rolle, welchen Webserver Sie verwenden. Offensichtlich kann der Debugger nicht verwendet werden, wenn PHP als CGI-Modul läuft. Als Apache- oder ISAPI-Modul gelingt die Installation problemlos. Damit ist zumindest unter Windows die Anwendung auf lokale Entwicklungscomputer beschränkt, denn aufgrund der Instabilität der ISAPI-Module ist die Kombination mit dem IIS für Produktionsumgebungen gänzlich ungeeignet.

Kein CGI!

10.2.2 Installation

Die folgende Beschreibung geht davon aus, dass sowohl Client als auch Server auf derselben Maschine laufen und das Windows 2000 als Betriebssystem verwendet wird. Desweiteren wird das ISAPI-Modul verwendet. Auf davon abweichende Angaben bei der Installation unter Linux wird hingewiesen. Mit Windows 9x hat der Autor keine Experimente angestellt, vermutlich läuft es nicht. Die Aussagen zu Linux dürften dagegen für Solaris auch zutreffen. Beachten Sie im Zweifelsfall die mitgelieferte Installationsanleitung.

Java-Installation

Die Java-Installation verlangt lediglich nach einer Pfadangabe für die installierten Dateien. Alles andere erledigt das Installationsprogramm.

Installation des Servers

Entpacken Sie die Dateien und kopieren Sie diese dann in ein Verzeichnis, wo Sie Programme ablegen, beispielsweise:

Konfiguration

```
c:\Programme\ZendIDE\Server
```

Beachten Sie, dass hier kein Installationsprogramm existiert.

Anschließend ist noch die PHP.INI anzupassen, damit die Erweiterung verfügbar wird. Ergänzen Sie dazu die folgenden Zeile:

```
zend_extension_ts="C:\Programme\ZendIDE\Server\ZendDebugger.dll";
```

Unter Unix verwenden Sie folgende Zeile:

```
zend_extension = "/usr/local/ZendIDE/Server/ZendDebugger.so";
```

Bei einem lokalen Entwicklungssystem können Sie die folgenden Schritte auslassen. Andernfalls ist es sinnvoll, den Zugriff auf den Server von bestimmten IP-Adressen zu unterbinden. Die folgende Zeile erlaubt bestimmten Hosts explizit den Zugriff:

```
zend_debugger.allow_hosts = 127.0.0.1, 192.168.100.10
```

Mit folgender Syntax geben Sie ein Netzwerk frei:

```
zend_debugger.allow_hosts = 192.168.0.0
```

Sie können dann wiederum einzelnen Adressen ausschließen:

```
zend_debugger.deny_hosts = 192.168.100.12, 192.168.100.13
```

Der Debugger benötigt eine Datei DUMMY.PHP im Stammverzeichnis des Webservers. Diese Datei kann leer sein, darf jedoch keinen Schreibschutz enthalten. Der Name kann in der ZendIDE (Clientprogramm) geändert werden, DUMMY.PHP ist die Standardeinstellung.

Lizensierung

Im nächsten Schritt muss der Server lizenziert werden. Dazu wird das Lizenzierungswerkzeug FLEXIm verwendet. Das Programm ist in der Lieferung des Pakets enthalten. Starten Sie es von der Kommandozeile:

```
c:\Programme\ZendIDE\Server>lmutil lmhostid -vsn
```

Abbildung 10.1:
Anforderung einer
Host-ID



Mit der angezeigten Nummer (DISK_SERIAL_NUM=20fc7ec0) können Sie die Lizenzdatei ZEND_DEBUGGER.DAT bei Zend anfordern. Für die 30-Tage-Testversion ist dies nicht notwendig, eine entsprechende Lizenzdatei liegt schon bei.

Unter Windows wird die Lizenz in der Registrierung eingetragen. Die entsprechende Anweisung steht in INSTALL_LICENSE.REG. Sie müssen diese Datei bearbeiten, wenn Sie ZendIDE nicht im Standardpfad installiert haben:

```
REGEDIT4
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session
Manager\Environment]
"ZEND_LICENSE_FILE"="C:\\Programme\\ZendIDE\\Server\\zend_debugger
.dat"
```

Der angegebene Pfad muss auf die Datei ZEND_DEBUGGER.DAT zeigen. Beachten Sie die anderen Pfade (hier nicht gezeigt) in der Registrierungsdatei. Wenn Sie gleichzeitig mit dem Encoder arbeiten, müssen Sie die Pfadangaben ebenso korrigieren bzw. erhalten, weil alle Angaben im selben Schlüssel registriert werden.

Doppelklicken Sie nun die Registrierungsanweisung – der Schlüssel wird eingetragen. Sie können dies nach Änderungen der Datei beliebig oft wiederholen.

Führen Sie nun einen ersten Test aus, indem Sie phpinfo aufrufen:

```
<?php
phpinfo();
?>
```

Testen der Installation

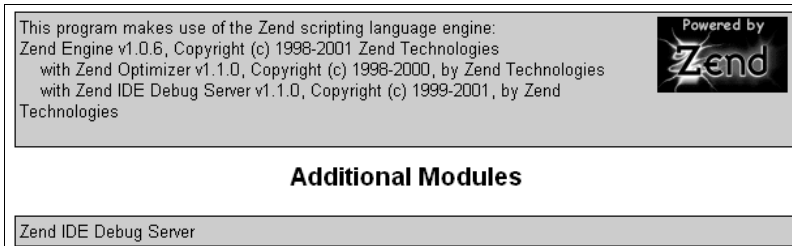


Abbildung 10.2:
Der Debugger wurde
erfolgreich aktiviert

Installation des Clients

Der Client wird mit einem Windows-Installationsprogramm geliefert, dass außer dem installierten Java JRE 1.3.1 keine weiteren Voraussetzungen benötigt. Die einzige Installationsoption bezieht sich auf die Dokumentation, die als PDF-Datei vorliegt.

10.2.3 Grundfunktionen

Nach der Installation von Client und Server sollten Sie einen einfachen Test mit der DIE ausführen, um ein Gefühl für die Arbeitsweise zu bekommen.

Einfacher Test der Arbeitsumgebung

Starten Sie die ZendIDE über START | PROGRAMME | ZENDIDE. Öffnen Sie die eine Datei und führen Sie diese über den Debugger aus. Dazu wählen Sie im Menü DEBUG die Option GO.

Konfiguration

Die Konfiguration erreichen Sie über **TOOLS | CUSTOMIZATION**. Einstellbar sind verschiedene Optionen für den Editor, den Debugger und die Gestaltung der Anzeige. Die interessantesten werden kurz vorgestellt.

Tooltip

Wenn der Debugger läuft, können Sie sich den Inhalt von Variablen anzeigen lassen, indem die Maus über den Namen geführt wird. Es gibt zwei Anzeigemodi dafür:

- **NORMAL**. Der Inhalt wird in einer blauen Box angezeigt:

Einige Zeichen

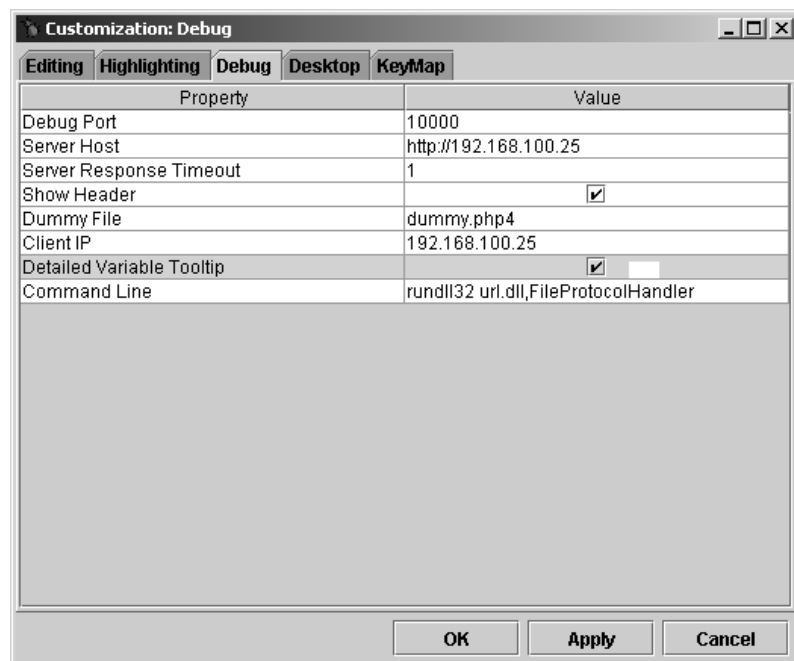
- **SERIALISIERT**. Der Inhalt erscheint so, wie es die PHP-Funktion `serialize` darstellt:

s:14:"Einige Zeichen";

Die Option kann folgendermaßen umgeschaltet werden:

1. Öffnen Sie die Konfiguration unter **TOOLS | CUSTOMIZATION**.
2. Wechseln Sie zur Registerkarte **DEBUG**.
3. Aktivieren Sie die Option **DETAILED VARIABLE TOOLTIP**.

Abbildung 10.3:
Aktivierung der
Anzeige aus-
führlicher Variablen-
inhalte



Autoindent

Eine andere interessante Funktion betrifft die automatische Einrückung nach einer öffnenden geschweiften Klammer. Entsprechend

erfolgt die Ausrückung wieder mit der schließenden Klammer. Quelltexte werden so besser lesbar.

Die Option kann folgendermaßen umgeschaltet werden:

1. Öffnen Sie die Konfiguration unter TOOLS | CUSTOMIZATION.
2. Wechseln Sie zur Registerkarte EDITING.
3. Wählen Sie in der Zeile AUTO INDENTATION ENABLED die Option TRUE.
4. Klicken Sie auf Optionen, um die Tiefe der Einrückung (Standard: vier Zeichen) und die Art (Leerzeichen oder Tabulatoren) einzustellen.

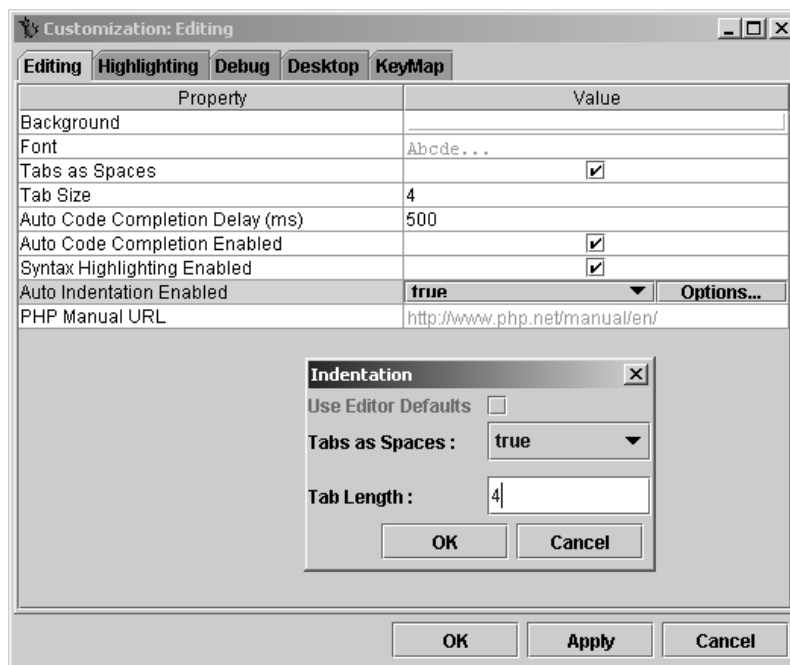
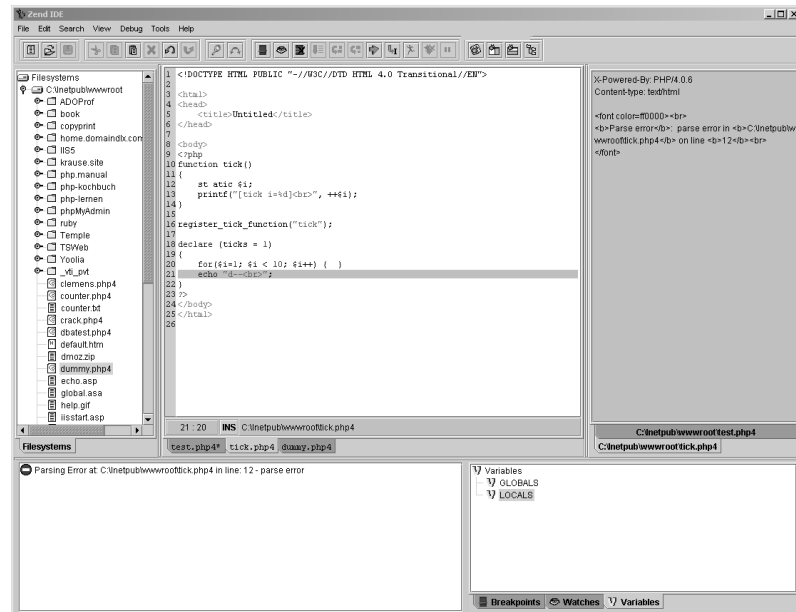


Abbildung 10.4:
Automatische
Einrückung an
geschweiften
Klammern

Umgang mit der IDE

Die IDE ist, zumindest in der hier getesteten Version 1.1, nicht sehr angenehm im Umgang. Offensichtlich treten die mangelnden Fähigkeiten von Java hier zu Tage, was derartige Applikationen betrifft. Ein paar Tipps sollen helfen, die ersten Schritte dennoch gehen zu können. Der Umgang lohnt eventuell wegen des integrierten Debuggers, auch wenn dieser das Niveau anderer kommerzieller Produkte bei weitem nicht erreicht – trotz des rasanten Preises von \$ 250.

Abbildung 10.5:
ZendIDE mit allen
Optionen



10.2.4 Die Fenster der IDE

Die IDE verfügt über mehrere Fenster, die sowohl zusammen (angedockt) als auch einzeln außerhalb des Hauptfensters angeordnet werden können:

- **Editorfenster**
Hier erfassen und bearbeiten Sie Quelltexte.
- **Dateibaum**
Hier haben Sie Zugriff auf ausgewählte Verzeichnisse.
- **Ausgabefenster**
Dieses Fenster enthält die Ausgaben des laufenden Skripts. Dabei wird der Quelltext angezeigt, so wie der Browser ihn erhält.
- **Meldungsfenster**
Hier werden Fehlermeldungen angezeigt, beispielsweise Parser-Fehler.

Umgang mit den Fenstern

Zwischen den Fenstern sind Verbindungslinien, an denen Sie mit der Maus die Größe verändern können. Wenn Sie die Kopfzeile eines Fensters anklicken und herausziehen, entsteht ein alleinstehendes und außerhalb des Hauptdialogs platzierbares Fenster. Um das Fenster wie anzudocken, führen Sie die Maus über den inneren Teil des Randbereiches (nicht jedoch über die Titelseite). Es erscheint – nur wenige

Pixel breit – ein dunkelgrauer Rahmen. Klicken Sie diesen an und ziehen Sie das Fenster dann wieder in den Teil des Hauptdialogs, wo es andocken soll.

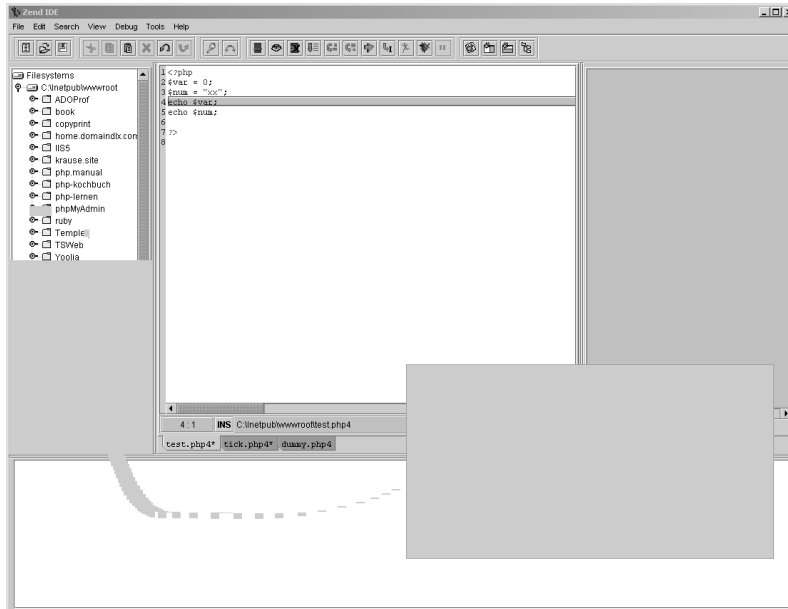


Abbildung 10.6:
Fenster im
Andockmodus
erscheinen als graue
Fläche (rechts unten)

Editorfenster

Im Editorfenster werden Skripte erfasst oder bearbeitet. HTML und PHP werden erkannt und mit unterschiedlichen Farben dargestellt, die durchgehend selbst einstellbar sind. Leider werden PHP-Funktionen nicht erkannt und es ist auch nicht möglich, hier eigene Listen zu erstellen. Nur reservierte Wörter werden, neben Zeichenketten und Zahlen, zuverlässig erkannt.

Die üblichen Tastenkombinationen zum Ausschneiden, Kopieren, Markieren und Einfügen stehen zur Verfügung.

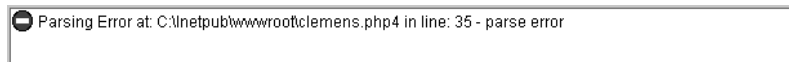
Ausgabefenster

Das Ausgabefenster zeigt alle Ausgaben des im Debugger laufenden Skripts an, ohne eine Darstellung wie im Browser vorzunehmen. Sie sehen hier auch Header und im Browser nicht sichtbar Informationen. Wenn der Debugger nicht intern, sondern über einen Browser aufgerufen wurde, bleibt das Fenster leer. Jedes startende Skript erzeugt ein weiteres Teilfenster. Klicken Sie mit der rechten Maustaste auf die Registerkarten mit den Dateinamen, um die Fenster einzeln wieder zu schließen.

Meldungsfenster

Das Meldungsfenster zeigt alle Notizen, Warnungen und Fehler an, die PHP erzeugt. Das funktioniert jedoch nur, wenn Sie keine eigene Fehlerbehandlung implementiert haben.

Abbildung 10.7:
Ausgabe im
Meldungsfenster

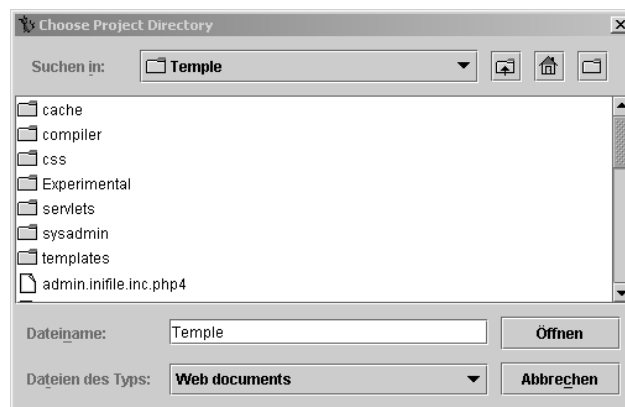


Dateibaum

Der Dateibaum startet entsprechend den Standardeinstellungen an der Root des Webservers oder dem Stammverzeichnis des aktuellen Benutzers. Dieser Baum ist jedoch nicht ein einfach durchsuchbarer Baum, sondern eine Liste von explizit ausgewählten Verzeichnissen. Klicken Sie mit der rechten Maustaste auf den Eintrag FILESYSTEMS und dann auf die Option ADD DIRECTORY. Sie können nun ein Projektverzeichnis der Liste hinzufügen und dort wie üblich navigieren.



Abbildung 10.8:
Auswahl eines
Projektverzeichnisses



Viele gravierende Mängel

Typische Projektfunktionen, wie FTP-Upload, Versionsvergleich und Zusammenfassung mehrerer Ordner zu einem Projekt fehlen völlig. Insofern ist die ZendIDE kaum mehr als ein einfacher Editor mit Debuggerfunktionen.

Viel kritischer ist aber zu sehen, dass der Zugriff auf Netzwerkpfade unter Windows nicht möglich ist, was die direkte Bearbeitung von Skripten auf einem Entwicklungsserver praktisch unmöglich macht.


10.2.5 Verwendung der Debuggerfunktionen

Da Editor und Projektverwaltung kaum das Niveau schlechter Shareware erreichen, stellt sich schnell die Frage nach dem Sinn der ZendIDE – vor allem in Anbetracht des enormen Preises.

Setzen von Unterbrechungspunkten

In Skripten können Sie, beispielsweise um bestimmte temporäre Zustände von Variablen zu untersuchen, Unterbrechungspunkte setzen – sogenannte Breakpoints. Dazu wählen Sie die Zeile aus, vor der angehalten werden soll. Dann können Sie den Unterbrechungspunkt setzen oder einen vorhandenen wieder entfernen:



- Mit dem Symbol  in der Symbolleiste
- Mit der Tastenkombination STRG-F8
- Über das Menü DEBUG, Option TOGGLE BREAKPOINT

Starten Sie des Debuggers

Sie können die Abarbeitung des Skripts dann im Debuggermodus starten. Ausgaben erscheinen im Ausgabefenster. Der Start kann auf diesem Weg erfolgen:

- Option GO , Menü DEBUG | GO.

Die Verarbeitung erfolgt bis zum Unterbrechungspunkt. Der Unterbrechungspunkt wird standardmäßig mit der Farbe Cyan hinterlegt. Die Position der Abarbeitung wird durch einen roten Rand um die betreffende Zeile markiert. Bei einem Stopp an einem Unterbrechungspunkt erscheint dieser also mit roten Rand.

```
29<?php
30$clemens="Meine Name ist Clemens";
31echo $clemens;
32 ?>
33</body>
```

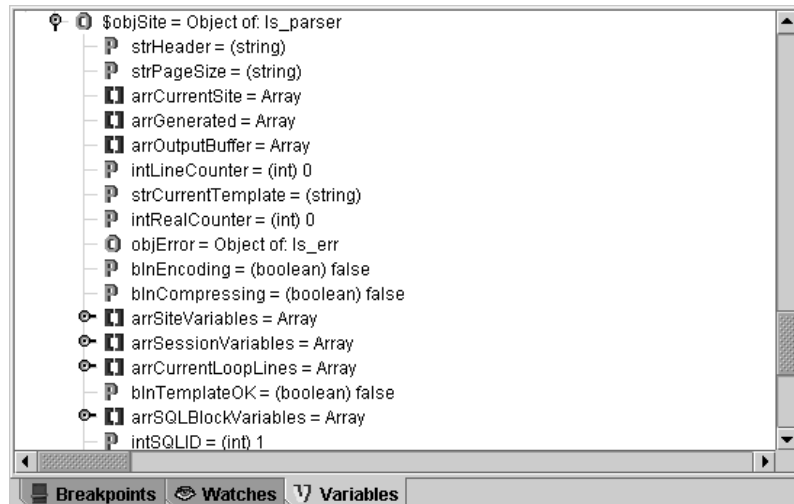
Abbildung 10.9:
Debugger-Stopp an
einem Unter-
brechungspunkt

Die Darstellung macht wie das gesamte Erscheinungsbild einen sehr unausgereiften Eindruck. Ebenso erschien bei Tests die Funktion nicht zuverlässig, manchmal übersieht der Debugger auch Unterbrechungspunkte.

Am Unterbrechungspunkt können sie im Variablenfenster dann den Zustand aller Variablen im Augenblick des Stopps einsehen. Dass ist unter Umständen sehr hilfreich.

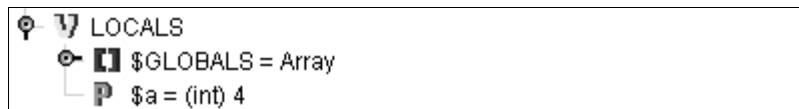
**Anzeige von
Variablen**

Abbildung 10.10:
Variablen am
Unterbrechungs-
punkt



Objekte werden auch angezeigt, allerdings sind nur die definierten Eigenschaftsvariablen zu sehen, keine Methoden. Befindet sich die Abarbeitung des Skripts gerade innerhalb einer Funktion, werden die dort gültigen Variablen im Zweig LOCALS angezeigt:

Abbildung 10.11:
Lokale Variablen








Die Verarbeitung des Skripts ohne Beachtung von Unterbrechungspunkten erfolgt mit dieser Anweisung:

- Option RUN , Menü DEBUG | RUN.

Schrittweise Abarbeitung von Skripten

Manchmal ist der Weg, den der Parser beim Verarbeiten von komplexen Skripten nimmt, von Interesse. Gehen Sie zur Verfolgung nach dieser Anleitung vor:

1. Setzen Sie einen Unterbrechungspunkt vor das erste `<?php`-Tag im ersten Skript.
2. Starten Sie die Abarbeitung mit GO. Der Debugger stoppt am Unterbrechungspunkt.
3. Setzen Sie nun die Abarbeitung schrittweise mit STEP INTO  fort (Menü DEBUG | STEP INTO bzw. F7).
4. Wenn Sie in eine Funktion oder Klasse hineingesprungen sind, kann die schrittweise Abarbeitung erheblich Zeit in Anspruch nehmen. Dann können Sie die Funktion bzw. Klasse mit STEP OUT

-  (Menü DEBUG | STEP OUT bzw. STRG-F7) wieder verlassen und hinter dem Funktionsaufruf fortsetzen.
5. Wenn Sie auf einem Funktionsaufruf stehen (der nächste Schritt also in die Funktion hineinspringen würde), können Sie die Anzeige mit STEP OVER  (Menü DEBUG | STEP OVER bzw. F8) übergehen. Die Funktion wird dann in der normalen Geschwindigkeit verarbeitet.
 6. Möchten Sie gezielt an einem bestimmten Punkt im Skript fortsetzen, klicken Sie mit der Maus den Cursor dorthin und wählen dann die Option GO TO CURSOR  (Menü DEBUG). Das Setzen des Cursors muss nach dem Start des Debuggers erfolgen, ansonsten wird der Punkt nicht erkannt.
 7. Stoppen Sie den Debugger jederzeit mit STOP DEBUGGER  (Menü DEBUG | STOP DEBUGGER). Am Ende eines Skripts stoppt der Debugger automatisch.

Der aktuelle Schritt wird mit einem roten Rahmen um die betreffende Zeile angezeigt. Die Ausgaben und Meldungen erscheinen so, wie die Verarbeitung des Skripts diese auslösen. Nach jedem Schritt können Sie den Inhalt einzelner Variablen oder der gesamten Variablenstruktur einsehen.

Abarbeitung von Skripten im Browser

Wenn die Ausgaben so erfolgen, dass eine Betrachtung im Ausgabefenster wenig sinnvoll erscheint oder eine Interaktion erforderlich ist, dann können Sie den Debugger auch über den Browser starten. Der Aufruf kann aus der ZendIDE heraus oder direkt im Browser erfolgen.

Aus der IDE heraus wählen Sie TOOLS | OPEN BROWSER. Wählen Sie dann die URL des Skripts aus und aktivieren dann das Kontrollkästchen START DEBUGGER.

Aufruf aus der IDE

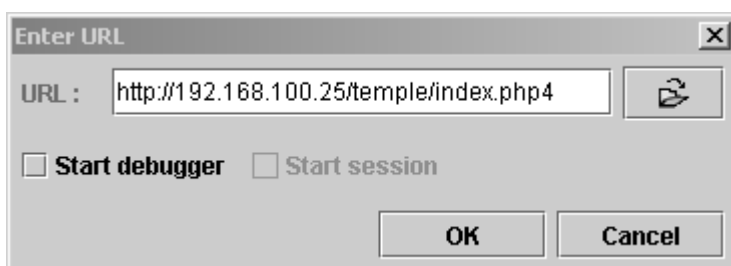


Abbildung 10.12:
Skript im Browser
abarbeiten

Der Browser startet. Im Editorfenster wird das betreffende Skript geladen, wo Sie wie üblich mit Unterbrechungspunkten arbeiten können. Das Ausgabefenster zeigt parallel alle Ausgaben an. Werden Dateien mit include eingebunden oder dort definierte Funktionen aufgerufen, werden diese jeweils einzeln im Augenblick der Abarbeitung ins Edi-

torfenster geladen und angezeigt. Auch hier können Sie, wenn zuvor am Eintrittspunkt eine Unterbechung definiert wurde, schrittweise vorgehen.

Direkter Aufruf des Debuggers

Die Interaktion des Skripts mit dem Debugger erfolgt über mehrere GET-Parmeter, die einfach an die URL angehängt werden:

- `start_debug=1`
1 startet den Debugger, 0 stoppt ihn wieder.
- `debug_port=10000`
Dies ist der Standardport des Debuggers.
- `debug_host=192.168.100.25`
Dies bezeichnet den Host, auf dem der Debugger läuft.
- `debug_start_session=1`
Startet eine neue Sitzung (optional).

10.3 Zend-Hilfsprogramme

Zend liefert einige Hilfsprogramme aus, die PHP sinnvoll ergänzen. Dabei ist die Verwendung an eine Installation auf dem Server gebunden, was bei vielen Providern nicht möglich ist.

10.3.1 Zend Optimizer



Der Zend Optimizer verbessert die Ausführungsgeschwindigkeit von Skripten. Er basiert auf der internen Arbeitsweise von PHP bzw. des Zend-Sprachkerns. Beim Abruf einer Seite erstellt der Compiler einen Zwischencode, der dann interpretiert wird. Dieser Zwischencode wird auch vom Zend Encoder verwendet, mit dem sich PHP-Code schützen lässt. Der Zend Optimizer »verbessert« diesen Zwischencode so, dass der nächste Verarbeitungsschritt etwas besser abläuft.

Verfügbarkeit

Zend Optimizer ist Freeware und über die Website von Zend erhältlich:

<http://www.zend.com>

Das Programm muss zu der eingesetzten PHP-Version passen. Wenn Sie einen Versionswechsel vornehmen, muss auch der Optimizer ausgetauscht werden.

Zend Optimizer unterstützt derzeit folgende Systeme:

Betriebssysteme

- Solaris ab Version 2.6
- Linux (auf i386)
- FreeBSD 3.4 und 4
- Windows NT 4 und Windows 2000

Das Programm wird als komprimierte Datei geliefert.

Installation unter Unix

Entpacken Sie zuerst das Paket:

```
$ gunzip -c <paket> | tar xvf-
```

Kopieren Sie dann das Modul ZENDOPTIMIZER.SO in einen Pfad, wo derartige Erweiterungen abgelegt werden, beispielsweise:

```
/usr/local/zend/lib/ZendOptimizer.so
```

Konfigurieren Sie nun die Datei PHP.INI, sodass der Optimizer aktiviert wird. Fügen Sie dazu folgende Zeilen hinzu (praktischerweise, aber nicht notwendigerweise in der Sektion LANGUAGE OPTIONS):

Konfiguration

```
zend_optimizer.optimization_level = 15;  
zend_extension = "/usr/local/zend/lib/ZendOptimizer.so";
```

Starten Sie nun den Apache-Webserver erneut, damit die PHP.INI gelesen wird, wenn Sie PHP als Modul verwenden.

Installation unter Windows

Entpacken Sie die ZIP-Datei und kopieren Sie das Modul ZENDOPTIMIZER.DLL, am besten dorthin, wo auch die anderen DLLs von PHP liegen:

```
c:\winnt\system32\ZendOptimizer.dll
```

Konfigurieren Sie nun die Datei PHP.INI, sodass der Optimizer aktiviert wird. Fügen Sie dazu folgende Zeilen hinzu (praktischerweise, aber nicht notwendigerweise in der Sektion LANGUAGE OPTIONS):

Konfiguration

```
zend_optimizer.optimization_level = 15;  
zend_extension_ts = c:\winnt\system32\ZendOptimizer.dll";
```

Wenn Sie die ISAPI-Erweiterung für den IIS verwenden, starten Sie den Webserver erneut:

```
c:>net stop w3svc  
c:>net start w3svc
```

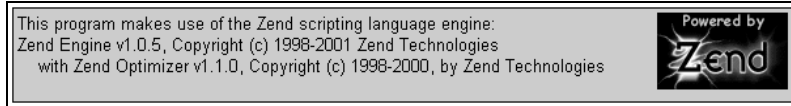
V
S
I
1
2
3
4
5
6
7
8
9

10

11
A
B
C
D
E

Kontrollieren Sie nun die erfolgreiche Installation durch Aufruf der Funktion `phpinfo`. Die Ausgabe enthält am Anfang im Block mit den Zend-Logo die Zeile »with Zend Optimizer v1.1.0,...«.

Abbildung 10.13:
Kontrolle der
Installation



Erweiterung mit dem Zend Encoder

Da der Optimizer auch die Ausführung geschützter Dateien erlaubt, steht dafür eine weitere Option in der `PHP.INI` zur Verfügung. Unterschiede zwischen Windows und Linux gibt es hier nicht. Standardmäßig untersucht der Optimizer alle Dateien auf kodierte Daten. Sie können dies unterdrücken, wenn der Encoder nicht verwendet wird, was die Ausführungsgeschwindigkeit nochmals etwas steigert. Fügen Sie dazu die folgende Zeile der `PHP.INI` hinzu:

```
zend_optimizer.enable_loader = 0;
```

10.3.2 Zend Cache



Das Programm ZendCache wird direkt in den Sprachkern integriert und speichert fertig gepackte Seiten zwischen, sodass spätere Abrufe schneller erfolgen. Die Steuerung der Eigenschaften kann über eine Browseroberfläche erfolgen.

Verfügbarkeit

ZendCache ist für folgende Betriebssysteme verfügbar:

- Linux (glibc 2.1)
- Red Hat 7.1
- Sparc Solaris 2.6 / 7 / 8
- FreeBSD 3.4 / 4.0.

Unterstützt werden die PHP-Versionen 4.0.6, 4.0.5, 4.0.4, 4.0.3. Als Webserver muss Apache 1.3.x eingesetzt werden.

10.3.3 Zend Encoder Unlimited



Ein weiteres interessantes Produkt ist der Zend Encoder Unlimited. Damit lassen sich fertige PHP-Skripte in den vom Interpreter verarbeiteten Zwischencode überführen. Die Verarbeitung setzt den ZendOptimizer voraus, der frei verfügbar ist. Aufwändig produzierte Programme lassen sich damit wirkungsvoll schützen. Das Produkt muss nur ein Mal für eine Entwicklungsumgebung erworben werden, kann

dann allerdings beliebig oft verschlüsselte Programme erzeugen. Der Preis ist mit \$ 2 400 erheblich und zielt auf kommerzielle Entwicklungsunternehmen.

Verfügbarkeit

Der Zend Encoder ist für folgende Betriebssysteme verfügbar:

- Linux (glibc 2.1)
- Red Hat 7.1
- Sparc Solaris 2.6 / 7 / 8
- FreeBSD 3.4 / 4.0
- Windows NT 4 / 2000

Unterstützt werden die PHP-Versionen ab 4.0.3. Als Webserver kann Apache 1.3.x , IIS 4/5, ZEUS und jeder CGI-kompatible Server eingesetzt werden.

10.3.4 Zend Launch Pad

Zend Launch Pad ist nur innerhalb der Zend Developer Suite erhältlich, die zugleich alle anderen bereits vorgestellten Pakete für eine Jahreslizenzgebühr von \$ 600 enthält. Mit dem Zend Launch Pad erleichtern Sie die Distribution fertiger PHP-Umgebungen zur Installation der für die Abarbeitung von Skripten notwendigen Programme und der passenden Konfiguration.

Über eine browserbasierte Oberfläche lässt sich die PHP.INI komfortabel einrichten. Des weiteren werden Informationen angezeigt über installierte Module, den verwendeten Webserver und die mitgelieferte Dokumentation zu PHP.





Praxis III – Das PHP-Projekt

In diesem Kapitel geht es um eine umfangreiche und professionelle Anwendung, deren Entwicklung von Grund auf gezeigt wird – *phpTemple* – ein Templatesystem. Mit diesem Programm können sehr große PHP-basierte Projekte besonders effizient entwickelt werden.

- phpTemple – Die Idee (Seite 769)
- Definition des Leistungsumfanges (Seite 772)
- Installation und Administration (Seite 800)
- Grundlegende Hinweise zum Prinzip (Seite 804)
- Anwendungsbeispiele (Seite 805)
- Realisierung: So funktioniert es! (Seite 808)
- phpTemple-Projekte im Web (Seite 840)
- phpTemple für professionellen Einsatz (Seite 845)

11.1 phpTemplate – Die Idee

Bei der praktischen Arbeit an größeren Projekten wird schnell klar, dass die Vermischung von HTML und PHP-Code die Wartung und Weiterentwicklung erschwert. Schon seit einiger Zeit sind verschiedene Lösungen im Umlauf, die häufiger zum Einsatz kommen.

11.1.1 Templates: Trennung von Code und Design

Klassisch werden für die Trennung von Code und Design Template-systeme eingesetzt. Dabei wird der HTML-Code in einer Datei abgelegt und die Programmieranweisungen in einer anderen. Durch spezielle Befehle werden die beiden Dateien dann beim Abruf durch den Benutzer miteinander verknüpft.

Vorteile

Vorteilhaft ist die einfache Trennung der Aufgabenbereiche bei der Herstellung der Daten. So können die HTML-Vorlagen durch Designer bearbeitet werden, ohne dass diese über PHP-Code stolpern. Noch problematischer ist das Verhalten verschiedener Editoren, die zwar exzellente Websites erstellen, mit Skriptcode aber nicht zurecht kommen.

Der Aufbau von kommerziellen Webseiten folgt einem bestimmten Schema, dass mit dem Prinzip des eingebetteten Codes kollidiert. So werden häufig komplette Layouts in Adobe Photoshop entworfen. Mit Hilfe von Image Ready werden diese Layouts dann in eine Rohseite zerlegt. Das Programm erstellt automatisch das Tabellengerüst der Seite sowie die Schaltflächen und Grafiken mitsamt der JavaScript-Effekte für die Menüpunkte. Am Ende entsteht mit wenigen Mausklicks eine fertige HTML-Seite, bestehend aus Tabellen und Grafiken.

Als Programmierer werden Sie in der Praxis mit solchen fertigen Seiten konfrontiert. Sicher ist es jetzt sehr einfach, den PHP-Code einzubauen und eine dynamische Website zu erhalten. Wenn sich nun am Design Änderungen ergeben, wird es kritisch. Praktisch müssen Sie Ihren Code nun wieder von Hand einbauen. Sicher lösen auch Templatesysteme das Problem nicht vollkommen – irgendwo muss dieser »Einbau« erfolgen – aber er vereinfacht das Verfahren.

Wie Templatesysteme funktionieren

Anstatt fertigen Code einzubauen, nutzen die Templatesysteme meist spezielle Makroanweisungen, die dann ihrerseits Programmcode aufrufen. Statt zeilenlanger Skripte oder vielfältiger Verflechtungen bleiben wenige Anweisungen zurück, die natürlich viel einfacher an-

zupassen und zu übertragen sind. Andere Systeme erzeugen HTML-Code über spezielle Funktionsaufrufe und reduzieren den austauschbaren HTML-Teil. Auch dies ist letztlich eine Vereinfachung durch Beschränkung des von Wartungsarbeiten betroffenen Teils.

Nachteile

Nicht verschwiegen werden soll, dass Templatesysteme auch Nachteile haben. So ist zum einen eine gewisse Einarbeitungszeit notwendig. Zusätzlich zu HTML und PHP kommen spezielle Anweisungen, die die Verknüpfung herstellen. Außerdem müssen sich Webdesigner und Programmierer auf ein System einigen – bei verteilter Arbeit nicht immer ganz einfach.

11.1.2 Templatesysteme für PHP

Bekannt geworden sind vier Templatesysteme, die häufig in PHP-Projekten zum Einsatz kommen:

- phpLIB
- Smarty
- FastTemplate
- IT[X] (im PHP-PEAR-Bereich zu finden)

Alle vier haben ihre spezifischen Vor- und Nachteile. Prägend ist jedoch immer die Dominanz des Codes.

Prinzipien von Templatesystemen

Generell kann man zwei Techniken unterscheiden, die zum Einsatz kommen können. Das eine Prinzip forciert den Aufbau der Seite und die Generierung des HTML-Codes über das Skript. Betrachten Sie den folgenden Code, wie er vom Templatesystem FastTemplate verwendet wird:

```
<?php
$tpl = new FastTemplate("pfad");
$tpl->assign(PAGETITLE, "Musterseite");
$tpl->parse(MAIN, "bask");
$tpl->FastPrint(MAIN);
?>
```

Der HTML-Code wird in einer Datei *bask.tpl* abgelegt. Er könnte folgendermaßen aussehen:

```
<HTML>
  <HEAD>
    <TITLE>{PAGETITLE}</TITLE>
  </HEAD>
```

```
<BODY>
  <H1>Willkommen</H1>
</BODY>
</HTML>
```

Der eigentliche Code kann natürlich so erweitert werden, dass die Variable in der HTML-Seite {PAGETITLE} dynamisch verändert wird. Ruft ein Benutzer diese Seite auf, startet FastTemplate, der Code wird abgearbeitet und die Variablen werden ersetzt. Die fertige Seite wird an den Browser ausgegeben.

Smarty geht noch einen Schritt weiter und erzeugt gleich fertige HTML-Schnippel. Im Prinzip unterscheidet es sich aber nicht von FastTemplate (abgesehen von vielen Zusatzfunktionen).

PHPLIB ist eine komplette Bibliothek, die auch auf diesem Ersetzungsprinzip basiert und zusätzlich viele Module liefert, die häufig benötigt werden, beispielsweise eine Benutzerauthentifizierung. Ihren Haupteinsatz fand sie jedoch zusammen mit PHP 3, wo sie das dort fehlenden Sessionmanagement nachbildete.

Alle drei Templatesysteme haben einen gravierenden Nachteil. So wird die Struktur der Seite, die Verflechtung der Skripte und damit die Initiierung der Projektentwicklung von PHP-Seite aus gesteuert. Wenn Sie mehrere Codefragmente benötigen, die innerhalb der Seite Schleifen bilden oder Datenbanken abfragen, wird der HTML-Code zerissen. Statt einer Vorlage arbeiten Sie mit mehreren Vorlagenschnippeln. Damit ist der Einsatz von Werkzeugen wie Image Ready aber praktisch unmöglich – die ihrerseits zwingend notwendig sind, um effizient und damit kommerziell nutzbar zu arbeiten.

Gravierende Nachteile

Das andere Prinzip: Kontrolle über HTML

Ein anderes Prinzip verlegt die Kontrolle über den Ablauf und die Verknüpfung in die HTML-Vorlage. Der PHP-Code rückt völlig in den Hintergrund und wird nur dort eingebunden, wo es komplizierte Berechnungen unumgänglich machen. Der Designer kann dann entscheiden, ob er einige einfache neue Tags lernt und damit selbst die notwendige Dynamik in die Seite bringt oder seine fertigen Seiten an einen Programmierer weitergibt.

Das oben gezeigte Beispiel für FastTemplate könnte dann folgendermaßen aussehen:

```
<SET VARIABLE="PAGETITLE" VALUE="Musterseite"/>
<HTML>
  <HEAD>
    <TITLE>{PAGETITLE}</TITLE>
  </HEAD>
  <BODY>
    <H1>Willkommen</H1>
```

```
</BODY>  
</HTML>
```

Programmiercode ist nicht mehr notwendig. Neu ist das Pseudotag SET, dass der Variablen einen Wert zuweist. Klar, dass Datenbankabfragen, Schleifen und Verzweigungen hier verfügbar gemacht werden müssen.

11.1.3 Die Idee: phpTemple

Der Autor hat sich nach intensiver Analyse der vorhandenen Systeme und im Rahmen eines größeren Projekts entschlossen, ein eigenes Templatesystem nach dem zweiten Prinzip zu schreiben. Es hat sich als ausgesprochen gut nutzbar erwiesen und wird inzwischen in Dutzenden kommerziellen Projekten eingesetzt.

Vorteile

Mit Hilfe von *phpTemple* konnte die Zeit zur Entwicklung einer Website um 80% verringert werden. Eine derartige Produktivitätssteigerung wurde mit anderen Systemen nicht erreicht. Der größte Effekt wurde erzielt, wenn Code und Design von zwei verschiedenen Personen entwickelt wurden. Der Anteil an Abstimmungsarbeiten wurde signifikant verringert.

Entwicklungsphasen

phpTemple ist noch in der Entwicklung. Der erreichte Stand steht dem praktischen Einsatz jedoch nicht entgegen. Das komplette Projekt wird hier mit dem Arbeitsstand Juli 2001 vorgestellt. Auch wenn Sie es nicht einsetzen können oder wollen, lohnt die Lektüre:

- Sie lernen viel über professionelle Techniken der Programmierung
- Einige Module sind auch unabhängig von *phpTemple* nutzbar

Der Code wird aus Platzgründen nachfolgend nur auszugsweise vorgestellt. Die wichtigsten Passagen sind jedoch hier zu finden. Der Code ist insgesamt gut kommentiert, sodass die Orientierung in den auf CD befindlichen Quellen auch gelingen sollte. Allerdings ist die komplette »Inline-Dokumentation« in Englisch gehalten. Eine deutsche Version ist nicht geplant. Das Produkt selbst ist dagegen weitestgehend sprachunabhängig.

11.2 Definition des Leistungsumfanges

phpTemple soll in der Praxis alle typischen Aufgaben erledigen, die bei der Programmierung kommerzieller Websites anfallen. Dazu gehören folgende Leistungen:

- Einfacher Abruf von Daten in Vorlagen
- Verwaltung der Vorlagen in einer Datenbank
- Vollständiges Formularmanagement
- Direkter Zugriff auf Daten aus Datenbanken in Vorlagen
- Weitestgehende Unabhängigkeit vom Datenbanktyp
- Integration einiger Basisfunktionen
- Installation ohne Eingriffe in den Code

Die einzelnen Aspekte werden nachfolgend vorgestellt.

Beachten Sie in den folgenden Darstellungen, dass alle Tags *in einer Zeile geschrieben* werden müssen, da der Parser zeilenweise arbeitet. Die Darstellung im Buch ist jedoch teilweise mehrzeilig, um die Lesbarkeit zu erhöhen.



Hinweis

11.2.1 Grundsätzlicher Aufbau

phpTemplate basiert auf einer festen Struktur, die an folgende Bedingungen gebunden ist:

- In einem beliebigen neuen Verzeichnis wird die INDEX.PHP platziert. Diese Datei sollte niemals verändert werden. Hier werden alle Module eingebunden und die Verarbeitung gestartet.
- Die gesamte Struktur wird darunter aufgebaut

Es müssen mindestens folgende Verzeichnisse unterhalb des Stammverzeichnisses vorhanden sein:

- \TEMPLATES

Hier liegen die Vorlagen, wobei diese in einer beliebig tiefen Struktur weiter geteilt werden können. Die Verwaltung der Vorlagen erfolgt über eine dynamische Linkauflösung in der Datenbank. Deshalb sind Änderungen an dieser Struktur sehr einfach ausführbar. Jede Vorlage generiert exklusiv eine Seite. Es wird von INDEX.PHP aufgerufen und ausgeführt.

- \SERVLETS

Hier liegen die Programmstücke, die sogenannten Servlets, die bestimmte Aktionen ausführen. Designer können Servlets mit speziellen Tags aufrufen, Daten übergeben und abholen, ohne die Programmiersprache oder die Programmlogik zu verstehen.

- \TEMPLATES\INCLUDE

In diesem Verzeichnis liegen – ohne weitere Linkauflösung über die Datenbank – weitere Vorlagen, die keine kompletten Seiten darstellen. Komplexe Vorlagen können durch Aufteilen auf eine Hauptvorlage (in \TEMPLATES) und mehrere Teilvorlagen (in \TEMPLATES\INCLUDE) vereinfacht werden. Außerdem können Teilvorlagen mehrfach verwendet werden.

Es ist empfehlenswert, folgende Verzeichnisse für alle anderen Daten zu verwenden:

- \CSS

Style Sheets, JavaScript-Dateien und andere Elemente für die Gestaltung.

- \DATA

Verzeichnis für Daten, hochgeladene Dateien usw.

- \IMAGES

Bilder, die von Vorlagen verwendet werden (Designbilder) und die zur Gestaltung der Site gehören.

- DATE\IMG

Bilder, die zu Daten gehören, beispielsweise aus der Datenbank heraus angesprochen werden. Wenn Sie einen Shop aufbauen, werden hier die Artikelbilder abgelegt.



Es ist möglich, alle Voreinstellungen für die Verzeichnisse in der Administration bzw. vor der Installation zu verändern. Die hier gezeigten Daten sind die mitgelieferten Voreinstellungen.

Aufruf von Vorlagen aus dem Include-Verzeichnis

Innerhalb jeder Vorlage kann an jeder beliebigen Stelle eine Vorlage aus dem Verzeichnis \INCLUDE eingeschlossen werden. Schreiben Sie an der entsprechenden Stelle einfach folgendes Tag:

```
<LS:INCLUDE TEMPLATE="name" />
```

In der Administration kann festgelegt werden, wie Sie die Dateien benennen. Der Standardwert der Dateierweiterung ist .HTM. Wenn Sie als Parameter des Attributes TEMPLATE *omnipresent* angeben, wird die Vorlage aus folgendem Pfad geladen:

```
<basedirphpTemple>/templates/include/omnipresent.htm
```

<basedirphpTemple> ist das Verzeichnis, in dem *phpTemple* (also der PHP-Code) abgelegt ist.

11.2.2 Variablen in Vorlagen

Der Zugriff auf Variablen in Formularen ist wenig spektakulär. Der Designer baut diese in der folgenden Form ein:

```
<title>${title}</title>
```

Dies hat den Vorteil, dass in der Seitenvorschau die Variablen stehen und das Design gut geprüft werden kann. Editoren stören sich an derartigen Zeichenfolgen nicht.

Vordefinierte Variablen

Einige Variablen sind bereits vordefiniert. Sie werden generell mit großen Buchstaben geschrieben.

- \${SESSIONID}

Die interne ID der Session. Für den Benutzer ist die Session-ID verschlüsselt.

- \${USERID}

Wenn die interne Benutzerverwaltung verwendet wird, steht die ID des aktuellen Benutzers in dieser Variablen.

- \${USERNAME}

Wenn die interne Benutzerverwaltung verwendet wird, steht der Anmelde-name des aktuellen Benutzers in dieser Variablen.

- \${LONGDATE}, \${LONGTIME}, \${SHORTDATE}, \${SHORTTIME}

Diese Variablen enthalten Datum und Uhrzeit in verschiedenen Ausgabeformen. Der Wert wird mit jedem Seitenaufruf aktualisiert.

- \${TEMPLATE}

In dieser Variablen steht immer der Name des aktuellen Templates in der nicht aufgelösten Form.

Systemvariablen

Während der Laufzeit entstehen weitere Variablen, die durch spezielle Präfixe eingeleitet werden:

- \${INFO_xxxx}

Diese Variablengruppe enthält Informationen über den Status der Daten bei der seitenweisen Ausgabe von Datenbankabfragen.

- `{NAVI_xxxx}`

Die Daten in diesen Variablen dienen dem Aufbau einer Navigation bei der seitenweisen Ausgabe von Datenbankabfragen.

- `{LINK_xxxx}`

Diese Variable weist auf den Link der Seite hin.

- `{MAIN}`

Das »Hauptskript«, das den Parser und damit *phpTemple* selbst nach jedem Seitenaufruf wieder startet.

Benutzerdefinierte Variablen

Sie können jederzeit innerhalb eines Templates Variablen erzeugen und wieder löschen. Dazu dienen zwei spezielle Tags:

- SET

Dieses Tag setzt eine Variable. Existiert diese bereits, wird der alte Wert überschrieben.

- UNSET

Dieses Tag löscht die Variablen endgültig.

Interessant ist der Einsatz in einer universellen Navigation. Ein solches Beispiel wird in ➡ Abschnitt 11.7 *phpTemple-Projekte im Web* ab Seite 840 beschrieben.

Globale Variablen (Applikationsvariablen)

PHP kennt keine Applikationsvariablen. Dies sind Variablen, die überall und jederzeit definiert werden können und die dann auf allen Seiten aller verbundenen Benutzer zur Verfügung stehen, ohne dass es einer erneuten Deklaration bedarf. Zwei Tags kommen dafür zum Einsatz:

- SETGLOBAL

Hiermit werden globale Variablen erzeugt. Sie bleiben bis zu einer expliziten Löschung mit UNSETGLOBAL erhalten.

- UNSETGLOBAL

Dieses Tag löscht globale Variablen wieder.

Schreiben Sie beispielsweise in der Administration Ihrer Site folgendes Tags in das entsprechende Formular:

```
<LS:SETGLOBAL VAR="Version" VALUE="{FIELD_version}"/>
```

Dann kann in allen Skripten aller Benutzer ohne jede weitere Zuweisung die Variable `${Version}` abgerufen werden, in der beispielsweise die Versionsnummer der Site steht.

Eine typische Anwendung ist ein Counter. Counter müssen in *phpTemple* nicht programmiert werden. Sie sind einfach da. Kombinieren Sie Applikationsvariablen mit einer Prozessanweisung: **Counter**

```
<LS:PROCESS INLINE="${counter} + 1" RETURN="counter"/>
<LS:SETGLOBAL VAR="counter" VALUE="${counter}"/>
Der aktuelle Stand des Zählers ist: ${counter}
```

11.2.3 Das Sitemanagement

phpTemple verfügt über eine datenbankgestützte Verwaltung von Vorlagen. Damit ist eine Trennung der Namen der Vorlagen von ihrem physischen Speicherplatz möglich. Globale Umbenennungen sind ebenso einfach möglich wie der temporäre Austausch zu Testzwecken. In beiden Fällen werden keinerlei Eingriffe in den Code notwendig. Designer können so eine Struktur der Seiten entwerfen, ohne ständige Rücksprache mit den Programmierern zu halten.

Aufruf von Vorlagen

Soll eine Vorlage aufgerufen werden, gibt es zwei Möglichkeiten, auf den Namen zuzugreifen. Zum einen sollte die Abbildung als Link möglich sein. Dazu wird folgende Variable eingebaut:

```
${LINK_templatename}
```

Dabei ist *templatename* der in der Datenbank hinterlegte Name einer Vorlage. Wo diese liegt und wie sie konkret heißt, ist hier nicht wichtig – dies wird alleine durch die Linkauflösung erledigt.

In Formularen kann –abhängig von der Situation – zu einer bestimmten Vorlage aus einer ganzen Auswahl verzweigt werden. Dies wird über spezielle Tags gesteuert, die im nächsten ➡ Abschnitt *Das Formularmanagement* beschrieben werden. Dort wird lediglich der Name angegeben:

```
... template="templatename" ...
```

Das Auflösen der Namen in physikalische Adressen findet über die Linkauflösung statt, siehe ➡ Abschnitt 11.4.1 *Prinzip der Linkauflösung* ab Seite 804.

11.2.4 Das Formularmanagement

Viel Aufwand verursacht das Management von Formularen. Ein typisches Beispiel ist die Wiederholung des Inhalts der Felder beim fehler-

V

S

I

1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E

haften Absenden der Daten. Es ist ausgesprochen lästig für den Benutzer, wegen eines einfachen Tippfehlers alle Daten erneut eingeben zu müssen. Aufwändig ist leider auch die Programmierung derartiger Funktionen in PHP.

Einrichtung des Formularmanagements

Damit das interne Formularmanagement funktioniert, muss phpTemplate das Ziel *aller* Formularaufrufe sein. Das `<form>`-Tag sieht deshalb immer folgendermaßen aus:

```
<form action="{MAIN}" method="post">
```

Alle übrigen in HTML zulässigen Attribute können ohne Einschränkungen verwendet werden.

Prinzip der Zielsteuerung

Es ist wichtig, dass der Designer der Site entscheiden kann, wohin ein Formular gesendet wird, also welche Vorlage als nächstes erscheint. Dazu wird ein spezielles neues Tag eingeführt, das die Steuerung bestimmt:

Syntax

```
<LS:FOLLOW CASE="type" [NAME="name"] TEMPLATE="template"/>
```

Das Tag FOLLOW verfügt über mehrere Attribute:

- CASE = "type"

Der Parameter *type* bestimmt den Fall, der mit diesem Tag definiert wird. Sie können folgende Parameter einsetzen:

- "ok"

Dieses Tag bestimmt das Ziel, wenn das Formular korrekt ausgefüllt wurde.

- "error"

Hiermit wird ein weiteres Ziel bestimmt, das bei fehlerhaft ausgefüllten Formularen aufgerufen werden soll.

- "submit"

In diesem Fall wird das Sprungziel in Abhängigkeit von einer Absende-Schaltfläche bestimmt. Sie können mehrere Absende-Schaltflächen definieren und damit mehrere Ziele erreichen.

- NAME

Dieses Attribut bestimmt den Namen der Absende-Schaltfläche, auf die die Definition angewendet wird. Es darf nur zusammen mit CASE="submit" eingesetzt werden.

- TEMPLATE

Der Name der Vorlage, die bei Eintreten des entsprechenden Ereignisses aufgerufen werden soll. Wenn sich das Formular selbst aufrufen soll, kann `${TEMPLATE}` eingesetzt werden:

```
TEMPLATE="${TEMPLATE}"
```

Eine typische Anwendung zeigt das folgende Beispiel. Dort werden drei Ziele definiert: Eines für das korrekt ausgefüllte Formular, eines für das fehlerhafte Formular und eines zur Realisierung eines Abbruchwunsches des Benutzers:

**Typische
Verzweigung**

```
<form action="${MAIN}" method="post">
  <LS:FOLLOW CASE="ok" TEMPLATE="saveform"/>
  <LS:FOLLOW CASE="error" TEMPLATE="${TEMPLATE}"/>
  <LS:FOLLOW CASE="submit" NAME="cancel" TEMPLATE="homepage"/>
  ...
  <input type="submit" name="send" value="Senden">
  <input type="submit" name="cancel" value="Abbrechen">
</form>
```

Was passiert hier? Das eigentliche Formular wurde hier noch nicht gezeigt. Trotzdem funktioniert der Code bereits – es ist keine einzige Zeile PHP notwendig. Treten im Formular Fehler auf, ruft sich das Formular selbst auf. Wurde es korrekt abgesendet (mit der Schaltfläche "send"), wird die Vorlage *saveform* gestartet. Klickt der Benutzer dagegen auf die Schaltfläche "cancel", wird *homepage* aufgerufen.

Datenwiederholung

Das erneute Aufrufen derselben Seite wurde bereits angesprochen. Es ist wichtig, die bereits erfassten Daten nicht zu verlieren. Deshalb werden alle Formularwerte in speziellen Variablen gespeichert. Um den Wert zu wiederholen, schreiben Sie Formularfelder folgendermaßen:

```
<input type="text" value="${FIELD_name}" name="name">
```

Feldvariablen stehen also unter dem Namen des Feldes zusammen mit dem Präfix `FIELD` zur Verfügung. Beim ersten Aufruf des Formulars sind die Werte leer. Mit Hilfe einer speziellen Anweisung können Sie einen Startwert festlegen:

```
<LS:IFEMPTY VAR="${FIELD_name}">
  <LS:SET VAR="FIELD_name" VALUE="Startwert"/>
</LS:IFEMPTY>
```

`IFEMPTY` und weitere Steueranweisungen werden in ➡ Abschnitt 11.2.6 *Steuerung in der Vorlage* ab Seite 792 vorgestellt.

Prüfanweisungen festlegen

Damit *phpTemplate* feststellen kann, ob Fehler auftraten, müssen Prüfanweisungen festgelegt werden. Sie können beliebig viele Felder prüfen lassen. Der Formularfehler, der zum Aufruf der mit CASE="error" definierten Vorlage führt, wird ausgelöst, wenn nur eines der Felder der Prüfung nicht standhält. Die Prüfanweisung wird direkt in die Felder integriert. Da HTML nicht über derartige Attribute verfügt, muss die Verarbeitung auf dem Server erfolgen. Dazu wird zuerst das Attribut `runat="server"` ergänzt:

```
<input runat="server"
      type="text"
      name="name"
      value="{FIELD_name}"/>
```

phpTemplate wird dieses Tag jetzt erkennen und verarbeiten. Damit dies auch zu bestimmten Aktionen führt, sind zwei weitere Attribute notwendig:

```
<input runat="server"
      type="text"
      name="name"
      required="MANDATORY"
      validationmsg="Dieses Feld muss ausgefüllt werden"
      value="{FIELD_name}"/>
```

Das Attribut `required` enthält die Prüfanweisung, `validationmsg` dagegen einen Fehlertext, der ausgegeben werden kann, wenn die Prüfung nicht bestanden wurde.

Spezielle Prüfanweisungen

phpTemplate kennt einige fertige Prüfanweisungen, die einfach durch Einsatz des Parameters in das Attribut `required` aktiviert werden:

- MANDATORY

Dieses Feld muss ausgefüllt werden, also mindestens ein Zeichen enthalten.

- EMAIL

Dieses Feld muss eine gültige E-Mail-Adresse enthalten.

- URL

Dieses Feld muss einen gültigen URL enthalten.

- DATE

Hier wird ein Datum im Standardformat der gewählten Sprache erwartet.

Alle anderen Einträge werden als regulärer Ausdruck interpretiert, wie er von der Funktion `preg_match` verarbeitet werden kann.

Die Fehlermeldungen stehen in Variablen zur Verfügung, können also einfach an jede beliebige Stelle auf der Seite platziert werden. Sie sind durch den Präfix `${ERROR_name}` erkennbar.

**Zugriff auf
Fehlerausgaben**

```
<table>
  <tr>
    <td><input type="text" runat="server" name="email"
      value="${FIELD_email}"
      required="EMAIL"
      validationmsg="E-Mail falsch oder nicht da"/>
    </td>
    <td>*</td>
    <td>${ERROR_email}</td>
  </tr>
</table>
```

Dieses Beispiel zeigt das Zusammenspiel der einzelnen Attribute. Auch hier ist tatsächlich keine Zeile PHP-Code zu schreiben oder anzupassen. Natürlich kann auch die Fehlermeldung selbst dynamisch gesteuert werden, indem eine Variable eingesetzt wird, beispielsweise für mehrsprachige Formulare.

Beachten Sie, dass die Verarbeitung der Kontrollwerte beim ersten Aufruf des leeren Formulars erfolgt. Auf der nächsten Seite oder nach dem Absenden werden diese Werte nur wiederholt.

Kontrollkästchen und Optionsfelder

Kontrollkästchen und Optionsfelder funktionieren ähnlich wie normale Formularfelder. Die Wiederholung der ausgewählten Option ist hier allerdings etwas anders gelöst, weil statt des Wertes des Attributs `value` das Attribut `checked` geschrieben werden muss. Entsprechend werden bei solchen Feldern Variablen mit einem weiteren speziellen Präfix erzeugt: `${CHECKED_name}` und `${RADIO_name_option}`. Da Optionsfelder in Gruppen auftreten können, in denen sie identische Namen haben, wird noch die Auswahl mit einbezogen:

```
<input runat="server" type="radio" name="script"
  value="php" ${RADIO_script_php} />
<input runat="server" type="radio" name="script"
  value="asp" ${RADIO_script_asp} />
<input runat="server" type="radio" name="script"
  value="php" ${RADIO_script_php} />
```

Analog funktioniert das bei Kontrollkästchen, hier ist der Name allerdings eindeutig:

```
<input runat="server" type="radio" name="php"
  value="1" ${CHECKED_php} />
<input runat="server" type="radio" name="asp"
  value="1" ${CHECKED_asp} />
```

```
<input runat="server" type="radio" name="perl"
      value="1" ${CHECKED_perl} />
```

Listensteuerung

Etwas größeren Aufwand verursachen Listenfelder. Vor allem die Wiederholung der Auswahl ist nicht einfach – insbesondere, wenn es sich um eine Mehrfachauswahl handelt. Beides wird mit *phpTemple* viel einfacher.

Statische Listen

Zuerst ein Beispiel, wie einfache statische Listen aufgebaut werden:

```
<select name="script" size="1" runat="server">
  <option ${SELECTED_script_php} value="php">PHP
  <option ${SELECTED_script_asp} value="asp">PHP
  <option ${SELECTED_script_perl} value="perl">PHP
</select>
```

Der Aufbau der Auswahlvariablen, die das Attribut `selected` erzeugt, ist `${SELECTED_name_option}`. Ob Einfach- oder Mehrfachauswahl, spielt hier keine Rolle.

Dynamische Listen

Dynamische Listen lassen sich auf zwei Wegen erzeugen: Zum Einen aus einer beliebigen Datenbankabfrage heraus. Im Vorgriff auf die im nächsten Abschnitt beschriebenen Datenbankzugriffe sieht dies in *phpTemple* folgendermaßen aus:

```
<select runat="server" name="script" size="1">
  <LS:DATA LOOPS="0" STMT="SELECT script, sname FROM liste">
    <option runat="server" ${AUTOSELECTED} option="${script}">
      ${sname}
    </option>
  </LS:DATA>
</select>
```

Es genügt hier, `${AUTOSELECTED}` einzufügen, damit nach dem erneuten Aufbau des Formulars mit den Werten aus der Datenbank die ursprüngliche Auswahl erhalten bleibt.

Automatische Listen

Listen werden sehr oft benötigt. *phpTemple* verfügt deshalb über eine automatische Listengenerierung, die über die Administration gesteuert werden kann. Der Designer kann die Listenwerte über ein HTML-Formular in der Administration erzeugen und diese dann folgendermaßen verwenden:

```
<select runat="server" list="sex" name="sex" size="2">
</select>
```

Daraus generiert *phpTemple* folgenden HTML-Code:

```
<select      name="sex" size="2">
  <option selected value="m">Männlich
```

```
<option value="w">Weiblich
</select>
```

Die Standardauswahl kann in der Datenbank festgelegt werden, danach wiederholt diese Liste die letzte Auswahl des Benutzers.

Zugriff auf Prüfinformationen

Falls in einer weiteren Vorlage auf die Prüfanweisungen zugegriffen werden muss, stehen auch diese zur Verfügung:

- `${REQUIRED_feldname}`

In dieser Variablen mit dem Präfix `REQUIRED` steht die Prüfanweisung.

- `${VALIDATIONMSG_feldname}`

In dieser Variablen steht die Fehlermeldung, unabhängig davon, ob Fehler auftraten oder nicht.

Hochladen von Dateien

Auch das Hochladen von Dateien ist normalerweise mit Programmcode verbunden. In *phpTemple* benötigen Sie davor ebenso wie bei anderen Formularelementen keinen Programmcode. *phpTemple* erkennt hochgeladene Dateien und sorgt selbst für die Ablage. Der folgende Code zeigt das Prinzip (denken Sie daran, dass das Tag auf eine Zeile geschrieben werden muss):

```
<form action="${MAIN}" enctype="multipart/form-data" name="load">
...
  <input type="file"
    name="bilder"
    runat="server"
    maxfilesize="50000"
    targetdir="uploads/${USERNAME}"
    accept="image/gif, image/jpg, image/jpeg"
  />
</form>
```

Das für das Hochladen erforderliche Tag wird also neben dem schon vorgestellten `runat="server"` um drei weitere Attribute ergänzt:

- `maxfilesize`

Hier geben Sie an, wie groß die Datei in Bytes maximal werden darf. Beispiel:

```
maxfilesize="50000"
```

**Attribute zur
Steuerung des
Hochladens**

- `targetdir`

Hiermit wird gesteuert, wo die Datei abgelegt wird. Der hier angegebene Pfad muss unterhalb des Stammverzeichnisses von *phpTemple* bereits existieren. Beispiel:

```
targetdir="uploads/${USERNAME}"
```

- `accept`

Dieses Attribut enthält eine durch Kommata getrennt Liste von MIME-Typen, die akzeptiert werden. Beispiel:

```
accept="image/gif, image/jpg, image/png, image/pjpeg"
```

Wenn ein Formular mit derartigen Feldern abgesendet wurde, erzeugt *phpTemple* im Ziel einige Variablen mit dem Präfix `UPLOAD`, die den Status der Aktion wiedergeben:

- `${UPLOADSIZEERROR_name}`

Fehlermeldung, wenn die Größe der Datei zu hoch war.

- `${UPLOADMIMEERROR_name}`

Fehlermeldung, wenn der angegebene Dateityp nicht stimmt.

- `${UPLOADERROR_name}`

Allgemeiner Hinweis, dass das Hochladen misslungen ist.

- `${UPLOADFULL_name}`

Vollständiger Name (mit Pfad) der hochgeladenen Datei.

- `${UPLOADNAME_name}`

Name (ohne Pfad) der hochgeladenen Datei.

- `${UPLOADDIR_name}`

Pfad (entspricht dem Attribut `targetdir`).

- `${UPLOADSIZE_name}`

Tatsächliche Größe der Datei.

- `${UPLOADMIME_name}`

Tatsächlicher (systembedingter) MIME-Typ der Datei.

In der aktuellen Version sind die Fehlermeldungen nicht in der Vorlage wählbar, sondern müssen in der Administration global eingestellt werden.

11.2.5 Zugriff auf Daten aus Datenbanken

Der Zugriff auf Datenbanken ist in größeren Projekten praktisch zwingend. *phpTemplate* ist ohnehin selbst auf eine Datenbank angewiesen. Entsprechend üppig fallen die Möglichkeiten aus, die hier geboten werden.

Generell erfolgt der Zugriff mit einigen speziellen Tags:

```
<LS:DATA></LS:DATA>  
<LS:DATANEXT/>  
<LS:DATAWHILE></LS:DATAWHILE>
```

Abfragen einer Datenbank

<LS:DATA> erlaubt mehrere Attribute, die jedoch nicht zwingend eingesetzt werden müssen:

- LOOPS

Dieses Attribut ist optional. Anzahl der Datensätze, die ausgegeben werden sollen. Der Wert 0 steht für unendlich.

- DELETE

Dieses Attribut ist optional. Folgende Parameter sind zulässig:

- "on"

Durch die DATA-Anweisung definierte Variablen werden am Austritt der Schleife wieder gelöscht.

- "off"

Definierte Variablen bleiben bis zum Seitenende erhalten.

- STMT

Dieses Attribut ist erforderlich. Es enthält die SQL-Abfrage. Beispiel:

```
STMT="SELECT * FROM adresse WHERE id = ${FIELD_id}"
```

- NUMROWS

Optionales Attribut. Name einer Variablen, in der die Anzahl der von der Abfrage generierten Datensätze gespeichert wird. Unabhängig von LOOPS.

- OFFSET

Optionales Attribut. Ein Offset, ab dem Datensätze gelesen werden. Wird zur Seitensteuerung mehrseitiger Listen verwendet. Beispiel:

```
OFFSET="${INFO_OFFSET}"
```

- ID

Dieses Attribut ist optional. Interne ID des Tags bei komplexeren Verschachtelungen und für DATAWHILE.

- INSERTID

Name einer Variablen, der die INSERT-ID nach einer SQL-Einfügeoperation zugewiesen wird (INSERT).

- NAVIGATION

Anzahl der Datensätze pro Seite, Erzeugt zugleich alle Variablen für eine seitenweise Ausgabe. Wenn das Attribut nicht angegeben wird, werden die Variablen auch nicht erzeugt.

Eine einfache Liste können Sie folgendermaßen erzeugen:

```
<ul>
<LS:DATA LOOPS="0" DELETE="on"
      STMT="SELECT name AS loopname FROM name">
  <LI>${loopname}
</LS:DATA>
</ul>
```

Tag-Arten

DATA existiert in zwei Formen: Als Container-Tag und als alleinstehendes Tag. Die Container-Form wird nur verwendet, um Schleifen zu bilden. Der einfache Aufruf generiert nur einen Datensatz, der dem aktuellen Satz an Variablen zugewiesen wird. Diese sind bis zum Ende der Vorlage gültig.

INSERT DELETE UPDATE

Es gibt bezüglich der Nutzung von SQL keine Einschränkungen. Speziell für INSERT ist das Attribut INSERTID interessant. Dessen Parameter bestimmt eine Variable, in der die ID der Einfügeoperation steht, was in Tabellen mit AUTO_INCREMENT-Spalten nutzbringend ist.

Für DELETE und UPDATE ist das Attribut NUMROWS wichtig, wo die Variable bestimmt wird, die die Anzahl der bearbeiteten Datensätze hinterlegt enthalten soll. Alle anderen Attribute haben keine Funktion und werden ignoriert.

Zugriff auf die Abfrageergebnisse

Die Ergebnisse einer SQL-Abfrage stehen direkt in den Variablen zur Verfügung, die den Spalten der Ergebnisliste entsprechen. Es ist unbedingt zu empfehlen, Aliase mit AS zu bilden. Der Designer kann sich die SQL-Anweisung möglicherweise selbst zusammenbauen. Ein typischer Aufbau ist dann:

```
<LS:DATA STMT="SELECT strasse AS strasse, ort AS ort FROM data"/>
```

In der Seite wird womöglich an vielen Stellen auf die Ergebnisse in der Form \${strasse} oder \${ort} zugegriffen. Wenn der Programmierer

entscheidet, seine Tabellen anders zu benennen – beispielsweise die Spalten mit *street* und *city* bezeichnet – würden viele Änderungen anfallen. Durch die Aliase muss nur die SQL-Anweisung an einer einzige Stelle verändert werden:

```
<LS:DATA STMT="SELECT street AS strasse, city AS ort FROM data"/>
```

Das ist eine weitere Verbesserung der Trennung der beiden an der Herstellung der Site beteiligten Gruppen.

Seitenweise Navigation

Längere Abfragelisten werden möglicherweise seitenweise ausgegeben. Praktisch ist dazu keine Programmierung notwendig. *phpTemple* erzeugt intern entsprechende Variablen:

- `${INFO_LINE}`
Steht immer zur Verfügung, auch wenn NAVIGATION nicht angegeben wurde. Hier ist die aktuellen Zeilennummer, mit 0 beginnend, gespeichert. Dieser Wert ändert sich also bei jedem Durchlauf.
- `${INFO_SUM}`
Anzahl der Seiten (die Anzahl der Elemente pro Seite wird durch den Parameter des Attributs NAVIGATION bestimmt).
- `${INFO_PAGESIZE}`
Anzahl der Elemente pro Seite (dies entspricht NAVIGATION und nicht unbedingt der tatsächlichen Anzahl auf der letzten Seite).
- `${INFO_OFFSET}`
Aktuelle Seite (!), mit 0 beginnend, schreibbar. Wird vor allem im OFFSET-Attribut von DATA eingesetzt.
- `${INFO_FROM}`
Nummer des ersten Elements der aktuellen Seite
- `${INFO_TO}`
Nummer des letzten Elements der aktuellen Seite
- `${NAVI_FIRST}`
Nummer der ersten Seite.
- `${NAVI_NEXT}`
Nummer der nächsten Seite.
- `${NAVI_PREVIOUS}`
Nummer der vorhergehenden Seite.

- `${NAVI_LAST}`

Nummer der letzten Seite.

Die folgenden Beispiele zeigen, wie Sie diese Variablen in Links, mit JavaScript und in Formularen einsetzen können.

Einsatz als Link

Das folgende Beispiel aus dem Projekt Mauerfotos (siehe ➡ Abschnitt 11.7.1 *Mauerfotos* ab Seite 841) zeigt, wie die Navigation eingesetzt werden kann.

```
<tr>
  <td colspan="2" align="right">
    <LS:IF NUM1="${INFO_SUM}" OP=">" NUM2="1">
      &nbsp;Fotos <b>${INFO_FROM}</b> bis <b>${INFO_TO}</b>
      von <b>${INFO_SUM}</b>
    </LS:IF>
    <LS:IF NUM1="${INFO_SUM}" OP="==" NUM2="1">
      Es wurde <b>1</b> Foto gefunden
    </LS:IF>
  </td>
  <td align="right">
    <LS:IF NUM1="${NAVI_FIRST}" OP="==" NUM2="${INFO_OFFSET}">
      
    <LS:ELSE/>
    
    </LS:IF>
    &nbsp;&nbsp;&nbsp;
    <LS:IF NUM1="${NAVI_PREVIOUS}" OP="=="
      NUM2="${INFO_OFFSET}">
      
    <LS:ELSE/>
    
    </LS:IF>
  </td>
  <td align="left">
    <LS:IF NUM1="${NAVI_NEXT}" OP="==" NUM2="${INFO_OFFSET}">
      
    <LS:ELSE/>
```

```

        
    </LS:IF>
    &nbsp;&nbsp;&nbsp;
    <LS:IF NUM1="${NAVI_LAST}" OP="==" NUM2="${INFO_OFFSET}">
        
    <LS:ELSE/>
        
    </LS:IF>
    </td>
    <td align="left"></td>
</tr>

```

Die Navigation ist auch in JavaScript problemlos einsetzbar. Der Parser ersetzt die Variablen in allen Bereichen der Seite. Derzeit wird jedoch nicht die Verarbeitung von externen JavaScript-Dateien unterstützt, die mit dem Attribut `src` eingebunden werden. Dadurch können aber Bereiche explizit von der Verarbeitung ausgenommen werden. Werden JavaScript-Bereiche, deren Verarbeitung notwendig ist, mehrfach verwendet, sollten sie als Einschlussdateien behandelt werden:

```
<LS:INCLUDE TEMPLATE="javascript"/>
```

In der Datei *javascript.htm* wird dann der JavaScript-Code abgelegt.

Die Kodierung der URLs im Text basiert auf der Erkennung der Attribute `href=`, `src=` oder `link=`. Wenn in JavaScript ein URL erzeugt oder verwendet wird, kann dies zu Problemen führen. Folgender Code funktioniert nicht:

```
win = open('${LINK_popup}', 'Titel');
```

Schreiben Sie dafür folgendes:

```
href="${LINK_popup}";
win = open(href, 'Titel');
```

Jetzt wird der Link erkannt und ersetzt.

**Einsatz mit
JavaScript**

**Probleme mit
JavaScript**

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Mehrspaltige Ausgabe

Ein Problem stellt oft die mehrspaltige Ausgabe dar. Der Wechsel der Spalten kann dabei nach zwei Prinzipien erfolgen: Entweder stehen die Werte der ersten Spalte untereinander, dann folgen die Werte der zweiten Spalte untereinander usw. oder der Aufbau erfolgt zeilenweise. Beide Methoden werden von *phpTemplate* unterstützt.

Ausgabe einer definierten Anzahl gleichlanger Spalten

Der folgende Code zeigt, wie eine vierspaltige Ausgabe programmiert werden kann:

```
<LS:SET VAR="columns" VALUE="4"/>
<LS:DATA STMT="SELECT COUNT(*) AS numcats,
                    CEILING(COUNT(*) / ${columns}) AS line,
                    CEILING(COUNT(*) / ${columns})*2 AS line2,
                    CEILING(COUNT(*) / ${columns})*3 AS line3
                    FROM rscout_cat "/>
<TR>
<TD class=a vAlign=top width="25%">
  <LS:DATA LOOPS="0"
    STMT="SELECT name AS catname, id AS catid
          FROM rscout_cat
          WHERE parent = 0 LIMIT 0, ${line}">
    ${catname}<br>
  </LS:DATA>
</TD>
<TD class=a vAlign=top width="25%">
  <LS:DATA LOOPS="0"
    STMT="SELECT name AS catname, id AS catid
          FROM rscout_cat LIMIT ${line}, ${line}">
    ${catname}<br>
  </LS:DATA>
</TD>
<TD class=a vAlign=top width="25%">
  <LS:DATA LOOPS="0"
    STMT="SELECT name AS catname, id AS catid
          FROM rscout_cat LIMIT ${line2}, ${line}">
    ${catname}<br>
  </LS:DATA>
</TD>
<TD class=a vAlign=top width="25%">
  <LS:DATA LOOPS="0"
    STMT="SELECT name AS catname, id AS catid
          FROM rscout_cat LIMIT ${line3}, ${line}">
    ${catname}<br>
  </LS:DATA>
</TD>
</TR>
```

Entscheidend ist die ersten Anweisung. Hier wird die Anzahl der Spalten festgelegt:

```
<LS:SET VAR="columns" VALUE="4"/>
```

Dann wird auf der Grundlage der tatsächlich vorhandenen Daten der Umfang der Spalten bestimmt, damit die Verteilung so gleichmäßig wie möglich erfolgt:

```
<LS:DATA STMT="SELECT COUNT(*) AS numcats,
                CEILING(COUNT(*) / ${columns}) AS line,
                CEILING(COUNT(*) / ${columns})*2 AS line2,
                CEILING(COUNT(*) / ${columns})*3 AS line3
                FROM rscout_cat "/>
```

Nun steht in `${numcats}` die Anzahl der Datensätze und in den Variablen `{line}`, `{line2}` und `{line3}` die Nummer des Datensatzes, mit dem die zweite, dritte und vierte Spalte beginnt. `{line}` enthält zugleich die Anzahl der Einträge pro Spalte. Die Ausgabe der Spalten erfolgt durch vier weitere Abfragen, die mit Hilfe der SQL-Anweisung `LIMIT` gesteuert wird.

Hier wird `<LS:DATANEXT/>` eingesetzt; ein Tag, mit dem der nächste Datensatz ermittelt wird. Das folgende Beispiel zeigt eine fünfspaltige Tabelle:

**Spaltenweise
Ausgabe mit
horizontalem
Wechsel**

```
<LS:DATA LOOPS="0" NAVIGATION="15"
    STMT="SELECT nr AS fotoid, title FROM mf_fotos">
  <tr>
    <td width="20%" valign="top" height="150">
      <br><font class="klein">${title}<br>${date}</font>
    </td>
    <LS:DATANEXT/>
    <td width="20%" valign="top" height="150">
      <br><font class="klein">${title}<br>${date}</font>
    </td>
    <LS:DATANEXT/>
    <td width="20%" valign="top" height="150">
      <br><font class="klein">${title}<br>${date}</font>
    </td>
    <LS:DATANEXT/>
    <td width="20%" valign="top" height="150">
      <br><font class="klein">${title}<br>${date}</font>
    </td>
    <LS:DATANEXT/>
    <td width="20%" valign="top" height="150">
      <br><font class="klein">${title}<br>${date}</font>
    </td>
  </tr>
</LS:DATA>
```

Wenn `DATANEXT` den letzten Datensatz erreicht hat, springt es hinter den nächsten (zu dieser Anweisung passenden) `DATA`-Tag. Der HTML-Code, der dazwischen steht, wird nicht ausgegeben. Gegebenenfalls

muss man dies mit IF anschließend korrigieren. Das ist aber in der Regel erst bei sehr komplizierten Tabellen notwendig.

Vorwegnahme von Ergebnissen

Manchmal werden die Ergebnisse einer Abfrage auf einer Seite mehrfach benötigt. Statt mehrerer Abfragen kann eine spezielle Form an den Anfang der Seite gestellt werden. Der erste Datensatz wird abgerufen und die Variablen stehen zur Verfügung. Irgendwo auf der Seite kann man darauf Bezug nehmen und den Rest in einer Schleife abrufen. Dazu wird das Tag DATAWHILE eingesetzt.

```
<LS:DATA LOOPS="0" NUMROWS="count" ID="3"
      STMT="SELECT name, id FROM adressen"/>
... Ausgaben von ${count}
<LS:DATAWHILE ID="3">
... Daten in der Schleife
</LS:DATAWHILE>
```

Wenn nur eine Anweisung verwendet wird, ist das Attribut ID optional, andernfalls muss es mit einer beliebigen Zahl > 0 versehen werden.

11.2.6 Steuerung in der Vorlage

Durch Verzweigungen kann der logische Fluss innerhalb einer Vorlage gesteuert werden. Normalerweise sollte dies eine Ausnahme sein. Es ist sinnvoller, die Seite logisch so zu strukturieren, dass möglichst wenige derartige Verzweigungen erforderlich sind. Besser ist, verschiedene Aktionen durch verschiedene Vorlagen erledigen zu lassen und gleiche Blöcke in diesen Vorlagen durch INCLUDE-Vorlagen aufzubauen.

Wenn dennoch situationsabhängige Entscheidungen notwendig sind, werden folgende Tags eingesetzt:

- <LS:IF>

Dieses Container-Tag vergleicht zwei Werte anhand eines Operators und führt den eingeschlossenen Teil aus, wenn die Bedingung Wahr ist.

- <LS:ELSE/>

Mit diesem Tag kann innerhalb des Containers ein alternativer Zweig gebildet werden. Außerhalb eines IF ist dieses Tag illegal.

- <LS:IFEMPTY>

Dieses Tag prüft, ob eine Variable leer ist und führt den eingeschlossenen Teil dann aus. Es kann auch mit ELSE kombiniert werden.

- <LS:IFNOT>

Dieses Tag entspricht IF, negiert aber die Bedeutung der Bedingung.

- <LS:STOP/>

Dieses Tag stoppt die Ausführung und bricht ab. Alle danach folgenden Tags werden ignoriert. *phpTemple* startet nur wieder, wenn bis dorthin ein Link oder ein Formular ausgegeben wurde, mit dem der Benutzer den Prozess wieder anstoßen kann.

Einsatz von bedingten Zweigen

IF und IFNOT benötigen immer genau drei Attribute:

- VAR1 oder NUM1

Dies ist der linke Teil der Bedingung. VAR1 behandelt den Wert als Zeichenkette, NUM1 dagegen als Zahl. Beispiel:

```
<LS:IF NUM1="{id}" OP="==" NUM2="0">
```

- OP

Dies ist der Operator. Hier können Sie == für Gleichheit, <=, <, >, >= oder != einsetzen.

- VAR2 oder NUM2

Dies ist der rechte Teil der Bedingung. VAR2 behandelt den Wert als Zeichenkette, NUM2 dagegen als Zahl.

Das folgende Beispiel zeigt, wie Sie eine Ausgabe steuern können:

Einsatzbeispiel

```
<LS:IF NUM1="{menge}" OP="==" NUM2="1">
  $menge Flasche
<LS:ELSE/>
  <LS:IF NUM1="{menge}" OP="==" NUM2="0">
    Keine Angebote
  <LS:ELSE/>
    {menge} Flaschen
  </LS:IF>
</LS:IF>
```

Die Anweisungen können beliebig tief verschachtelt werden und alle anderen Tags enthalten. Der nicht verarbeitete Teil wird völlig ignoriert – auch *phpTemple*-Tags werden nicht ausgeführt.

Leere Variablen erkennen

Bei IFEMPTY wird nur eine Variable angegeben. Die Bedingung ist Wahr, wenn die Variable leer ist. Einziges Attribut ist VAR. Mit ELSE kann eine gefüllte Variable erkannt werden:

```
<LS:IFEMPTY VAR="{FIELD_email}">
    Sie müssen die E-Mail-Adresse ausfüllen.
<LS:ELSE/>
    Vielen Dank für die Angabe ({FIELD_email}).
</LS:IFEMPTY>
```

11.2.7 Zugriff auf PHP

Manchmal kommt man nicht umhin, doch reinen PHP-Code auszuführen. Selbstverständlich passiert dies nicht innerhalb der Vorlagen. Für ganz einfache Berechnungen gibt es jedoch einen vereinfachten Weg, der Berechnungen ohne Programmierung erlaubt. In jedem Fall wird dazu das Tag PROCESS eingesetzt.

Interne Berechnungen

Für interne Berechnungen wird PROCESS mit folgenden Attributen verwendet:

- INLINE

Hier steht der Code der Berechnung.

- RETURN

Hiermit wird der Name einer Variablen angegeben, die das Ergebnis enthält. Der Wert muss ein Skalar sein. Zusammengesetzte Datentypen werden nicht unterstützt.

Diese Funktion nutzt die Datenbank für die Prozessverarbeitung, nicht PHP. Damit ist eine bessere Skalierung möglich. Ein Beispiel:

```
<LS:PROCESS INLINE="3*7" RETURN="result"/>
Das Ergebnis lautet: {result}
```

Intern wird folgende SQL-Anweisung erzeugt und ausgeführt:

```
SELECT 3*7 AS result FROM dummy
```

Die Tabelle *dummy* wird nur zur Bedienung der Syntax benötigt.

Externe Programme

Alle externen Programme werden über eine bestimmte Schnittstelle bedient und stehen als sogenannte Servlets zur Verfügung. Ihnen können direkt aus der Vorlage heraus Parameter übergeben werden. Durch die streng definierte Schnittstelle ist es möglich, dass Program-

mierer und Designer völlig unabhängig voneinander entwickeln. Sie müssen sich nur auf den Übergabepunkt einigen.

Für externe Programme benötigt das Tag `PROCESS` folgende Attribute:

- `SERVLET`
- `RETURN`

Alle anderen Attribute, die gefunden werden, interpretiert *phpTemple* als Übergabepunkte und stellt die Werte dem Servlet zur Verfügung:

```
<LS:PROCESS SERVLET="img" RETURN="tag" SRC="{url}"  
W="23" H="2"/>
```

Hier wird das Servlet *img* ausgeführt, dessen Ergebnis in `{tag}` steht. Im Servlet stehen die Werte der übrigen Attribute zur Verfügung. Da es sich hier nun um PHP handelt, werden diese Daten als Array übergeben:

```
$SV['SRC'];  
$SV['W'];  
$SV['H'];
```

Servlets laufen im Kontext einer Funktion ab, und zwar innerhalb der Klasse *LS_parser*. Damit ist klar, dass sie sich so verhalten müssen, dass der Parser nicht gestört wird. Einschränkungen gibt es bezüglich der Definition eigener Funktionen und Klassen. Dieses Verhalten wird in einer künftigen Version von *phpTemple* verbessert.

**Bedingungen für
Servlets**

11.2.8 HTML-spezifische Funktionen

Einige Funktionen, die häufig benötigt werden, erleichtern die Gestaltung von Webseiten erheblich. *phpTemple* enthält erweiterbare Makrofunktionen für folgende Bereiche:

- Bilderzeugung
- Headererzeugung (beispielsweise für WML)
- Kompression und Manipulationsschutz
- Cachefunktionen

Bilderzeugung

Für die Bilderzeugung wird das Tag `<LS:IMG>` verwendet. Es erzeugt an der gegebenen Stelle ein äquivalentes ``-Tag mit einem Aufruf der Standardbibliothek. Die Parameter sind verschlüsselt, sodass die Bilderzeugung nur sehr schwer manipuliert werden kann. Die gesamte Steuerung wird mit mehreren Attributen vorgenommen:

- WIDTH

Bestimmt die Breite des Bildes, sowohl physisch, bei der Erzeugung, als auch in Form des `width`-Attribut in HTML.

- HEIGHT

Bestimmt die Höhe des Bildes, sowohl physisch, bei der Erzeugung, als auch in Form des `height`-Attribut in HTML.

- FONTFACE

Hiermit wird der Font festgelegt. Dieser wird immer in Kleinschreibung gewandelt. Auf UNIX-Systemen muss der hinterlegte Font deshalb kleingeschrieben sein. Es werden nur TrueType-Fonts akzeptiert, die auf `.TTF` enden. Die Erweiterung wird automatisch angehängt, der Pfad zu den Fontdateien kann in der `LS.INI` bestimmt werden. Er muss relativ zum Stammverzeichnis von *phpTemple* sein, Windows-Fonts müssen möglicherweise umkopiert werden. Alternative Font-Angaben sind nicht zulässig. Der gewählte Font muss existieren, andernfalls wird kein Bild erzeugt.

- FONTSIZE

Die Höhe des Fonts in Punkt. Ein Punkt entspricht $\frac{1}{72}$ Zoll oder 0,35 mm. Bei einer Standardauflösung des Monitors von 72 dpi gilt also 1 Punkt = 1 Pixel. Dies kann jedoch je nach Bilddarstellung auch etwas davon abweichen.

- ROTATE

Dieser Parameter bestimmt einen Winkel in Grad, um den der Text gedreht wird. Der Drehpunkt ist das Zentrum des Textes (dies entspricht nicht dem Verhalten der `image`-Funktionen in PHP, die an der linken unteren Ecke drehen). Die Richtung ist entgegen dem Uhrzeigersinn. Folgende Anweisung lässt Text von unten nach oben laufen:

```
ROTATE="90"
```

- COLOR

Hiermit wird die Farbe des Textes bestimmt. Zulässig sind nur hexadezimale Farbangaben ohne führendes `#`-Zeichen im RGB-Format.

- BGCOLOR

Dieses Attribut bestimmt die Hintergrundfarbe. Zulässig sind nur hexadezimale Farbangaben ohne führendes `#`-Zeichen im RGB-Format.

- BACKGROUND

Anstatt einer Hintergrundfarbe kann auch ein Bild angegeben werden. Es sollte in der Größe passend zum erzeugten Bild sein, ansonsten wird es von links oben beginnend ausgeschnitten.

- BORDER

Bestimmt den Rand, der jedoch nicht im Bild erzeugt wird, sondern lediglich ein äquivalentes border-Attribut erzeugt.

- BORDERCOLOR

Bestimmt die Randfarbe; der Rand wird jedoch nicht im Bild erzeugt, sondern lediglich ein äquivalentes bordercolor-Attribut erzeugt.

- CLASS, STYLE, ID

Diese Attribute werden unverändert durchgereicht.

- TEXT

Diese ist der Text, der angezeigt werden soll. Wenn er zu lang ist, läuft er aus dem Bild heraus, ohne dass ein Fehler erzeugt wird. Variablen können eingesetzt werden. Beispiel:

```
TEXT="Sie sind der ${counter}. Besucher"
```

- ALIGN

Die Textausrichtung. Erlaubt sind als Parameter "left", "right" und "center". Wenn der Winkel (ROTATE).

```
<LS:SET VAR="pagename" VALUE="Suchen"/>
<LS:IMG WIDTH="640" HEIGHT="50" TEXT="${pagename}"
  FONTFACE="verdanab" FONTSIZE="14" COLOR="000000"
  BGCOLOR="87CEEB" ALIGN="center" BORDER="1"/>
```

Beispiel

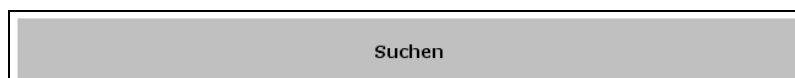


Abbildung 11.1:
Ausgabe des Beispiels

In der LS.INI sind einige Standardwerte festgelegt. Fehlen die Attribute, werden die Standardwerte eingesetzt. Sie finden diese Parameter in der Sektion [function defaults]:

Standardwerte

```
DefaultImageBack    /img/default.jpg
DefaultImageWidth    468
DefaultImageHeight   80
DefaultImageBorder   0
DefaultImageFont     arial
DefaultImageSize     14
FontPath             css/
```

Nach einer Änderung muss das Installationsskript neu gestartet werden.

Header

Mit dem Header-Tag werden spezielle Kopfzeilen der HTTP-Nachricht erzeugt. Eine Einsatzmöglichkeit ist die Programmierung von WML-Seiten für WAP-Handys.

WML

WML wird mit einem speziellen Header erzeugt. Den Rest erledigt *phpTemple*.

```
<LS:HEADER TYPE="wml"/>
<wml>
  <card>...</card>
</wml>
```

Kompression, Kodierung und Manipulationsschutz

Kompression

HTML-Seiten enthalten durch Einrückungen und Zeilenumbrüche ungefähr 30% Zeichen (sogenannte Whitespaces), die der Browser zur Anzeige nicht benötigt. Mit der Kompressionsfunktion werden diese Leerzeichen vor der Auslieferung der fertigen Seite entfernt. Auf die Vorlagen hat dies keinen Einfluss. Die Kompression kann auf der Seite jederzeit ein- und ausgeschaltet werden. Folgende Anweisung schaltet die Funktion ein:

```
<LS:ENCODING COMPRESS="on"/>
```

Entsprechend wird die Kompression folgendermaßen abgeschaltet:

```
<LS:ENCODING COMPRESS="off"/>
```

Es ist empfehlenswert, den Eintrag in einer Einschlussvorlage vorzunehmen.

Kodierung

Oft ist es lästig, bei der Erfassung von HTML die Umlaute richtig einzugeben. Editoren erledigen diese Umwandlung zwar automatisch – die Texte werden im Quelltext aber schwerer lesbar. Schwerer ist die Umwandlung von Daten aus Datenbanken oder fremden Quellen – hier sind aufwändige Filter einzubauen. *phpTemple* erledigt dies automatisch. Für die gesamte Seite oder Teile davon lassen sich Umlaute automatisch konvertieren. Wenden Sie das Tag `ENCODING` folgendermaßen an:

```
<LS:ENCODING SWITCH="on"/>
```

Mit `SWITCH="off"` kann die Umwandlung jederzeit wieder ausgeschaltet werden.

Das geht so nicht, denn die Kodierung bezieht zwar den Parameter *newsid* mit ein, nicht jedoch die Variable *id*. Korrekt wäre folgende Zeile:

```
window.location.href='${LINK_admin_list_news}'+&newsid='+id;
```

Damit endet die Kodierung vor den clientseitig erzeugten Parametern. Deren Kodierung ist nicht sinnvoll, weil der JavaScript-Code ohnehin einsehbar ist.

Leistungssteigerung: Komfortable Cache-Funktion

PERSISTENT speichert die komplett geparste Seite zwischen. Beim nächsten Aufruf werden die Parameter erneut ausgewertet. Ist die Seite im Cache noch aktuell, wird der Parser abgebrochen und die fertige Seite sofort an den Browser gesendet.

Folgende Attribute sind anzugeben:

- EXPIRES

Dieses Attribut ist erforderlich und bestimmt das Ablaufdatum im Cache. Dauerhafte Seiten müssen über die Administration freigegeben werden. Alternativ kann auch der Wert wieder im Template geändert werden, weil dieser immer geparkt wird. Entweder eine absolute Angabe im Format YYYY/MM/DD hh:mm:ss oder eine relative in Minuten +MMMM (erkenntlich am führenden +).

- REALM

Optionales Attribut, das folgende Parameter erhalten kann:

- "peruser"

Speichert eine Kopie pro Benutzer unter der UserID.

- "persession"

Speichert eine Kopie pro Session. Wird am Ende der Session gelöscht.



Hinweis

Diese Funktion ist noch in Entwicklung und funktioniert in der hier vorgestellten Version 0.9 noch nicht zufriedenstellend.

11.3 Installation und Administration

phpTemple ist sehr einfach zu installieren. Zuerst muss natürlich PHP 4 und MySQL problemlos laufen. PHP 4 kann ab Version 4.0.5 eingesetzt werden. Zur ersten Installation wird die Datei `INSTALL.HTML` aufgerufen. Dort kann die Datei `LS.INI` bearbeitet werden. Nach dem Absenden wird eine Konfigurationsdatei erzeugt, die Tabellen werden abgelegt und *phpTemple* startet mit der Stammvorlage und den mit-

gelieferten Übungsdateien. Diese Installation muss nur einmal vorgenommen werden.

11.3.1 Aufbau der Konfigurationsdatei

Die Konfigurationsdatei besteht aus Sektionen, die mit [section] bezeichnet sind und Einträgen, die jeweils aus einem Schlüsselwort und einem Wert bestehen. Kommentarzeilen beginnen mit einem Semikolon. Semikola am Ende einer Zeile leiten ebenfalls Kommentare ein. Die Verarbeitung der Konfigurationsdatei LS.INI erfolgt, ebenso wie die jeder anderen »persönlichen« Konfiguration, mit Hilfe spezieller Tags in der Administration.

Die Konfigurationsdaten

Die hier gezeigte Datei gilt für Version 0.9. Die Daten werden während der Installation in die Konfigurationsdatei CONFIG.INC.PHP4 geladen.

```
[database]
dbType           = mysql      ; dbase-type
dbServer         = localhost
dbDatabase       = rscout
dbUser           = root
dbPass
dbPort           = 3306       ; Port for mysql
[structure]
ParserFile       = index.php4 ; Name of the parser file
TemplateHome     = index.htm  ; Name of start template
TemplateDir      = templates/ ; Name of the template dir
TemplateInclude  = include/
ServletDir       = servlets/
AdminDir         = sysadmin/
AdminHome       = admin.htm   ; administration
AdminInclude     = includes/
TemplateCache    = cache/
Language        = de
[naming]
NameSpace        = rscout     ; Namespace
PExtension       = .php4     ; Extension of PHP-Files
TExtension       = .htm      ; Extension of Templates
IExtension       = .htm      ; Extension of Includes
SEExtension      = .php4     ; Extension for Servlets
[function defaults]
DefaultImageBack = /img/default.jpg
DefaultImageWidth = 468
DefaultImageHeight = 80
DefaultImageBorder = 0
DefaultImageFont = Arial
```

```
DefaultImageSize    = 14    ; Always uses 'pt' as measure
FontPath            = css/
```

11.3.2 Administration

Die Administration erlaubt den Zugriff auf die Systemtabellen und Einstellungen, die bei laufendem Betrieb geändert werden können. In der Version 0.9 ist diese noch nicht vollkommen fertiggestellt und wird hier nicht weiter betrachtet. Sie können alle Parameter auch mit *phpMyAdmin* vornehmen, wenn auch mit weniger Komfort.

Aufbau der Konfigurationsdatei

Die Konfigurationsdatei ist eine einfache Textdatei, die mehrere Abschnitte enthält. Aufbau und Verwendung entsprechen der `PHP.INI` von PHP. Änderungen wirken sich allerdings nur aus, wenn die Installation erneut gestartet wird. Daten in Kundentabellen werden dabei nicht verändert oder gelöscht.

Der erste Abschnitt steuert die Datenbankzugriffe. Zuerst wird der Typ der Datenbank eingestellt, der Standardwert ist `mysql`:

[database]

```
dbType              mysql
```

Dann folgt der Name des Datenbankservers:

```
dbServer            localhost
```

Der nächste Wert enthält den Namen der Datenbank:

```
dbDatabase          berlinshop
```

Die nächsten beiden Parameter enthalten Name und Kennwort für Datenbank:

```
dbUser              root
dbPass
```

Falls die Datenbank unter einem speziellen Port erreichbar ist, wird dieser folgendermaßen angegeben:

```
dbPort              3306
```

[structure]

Der folgende Abschnitt bildet die gewählte Verzeichnisstruktur ab. Zuerst wird der Name des »Stammskripts« angegeben. Normalerweise sollte dieser Wert nicht verändert werden:

```
ParserFile          index.php4
```

Dann folgt der Name der Standardvorlage und des Vorlagenverzeichnisses sowie des Unterverzeichnisses für die Einschlussdateien:

```
TemplateHome        index.htm
TemplateDir          templates/
TemplateInclude      templates/include/
```

Es folgt der Name des Verzeichnisses, in dem die Servlets (PHP-Programme) gespeichert sind:

ServletDir	servlets/
------------	-----------

Die integrierte Administration arbeitet selbst mit *phpTemple*-Vorlagen. Das Stammverzeichnis und Startvorlage können konfiguriert werden:

AdminDir	admin/
AdminHome	admin.htm

Die Administration ist mehrsprachig, ebenso einige Basisfunktionen. Auch für die Verwendung in mehrsprachigen Umgebungen kann die Standardsprache eingestellt werden:

Language	de
----------	----

Für die Cache-Funktion kann ein weiteres Verzeichnis angegeben werden, dass die fertigen Seiten aufnimmt. Dort müssen Schreibrechte gegeben werden.

TemplateCache	cache/
---------------	--------

Damit in einer Datenbank mehrere *phpTemple*-Systeme laufen können, **[naming]** wird mit einem globalen Präfix gearbeitet:

NameSpace	bs
-----------	----

Je nach Version und Einrichtung des Webserver kann die Dateierweiterung angepasst werden. Folgender Eintrag gilt für die Skript-Dateien selbst:

PExtension	.php4
------------	-------

Die Benennung der Vorlagen ist beliebig, ebenso die der Einschlussdateien:

TExtension	.htm
IExtension	.htm

Servlets sollte können ebenso beliebig benannt werden, ausgeführt werden sie im Kontext des Parameters PExtension:

SExtension	.php4
------------	-------

Die Funktionsbibliothek umfasst einige häufig benötigte Makrofunktionen – vor allem zur Bilderzeugung. Deren Grundeinstellungen finden Sie im folgenden Abschnitt (Beachten Sie die Kommentare):

DefaultImageBack	/img/default.jpg	; Hintergrundbild
DefaultImageWidth	468	; Breite
DefaultImageHeight	80	; Höhe
DefaultImageBorder	0	; Randbreite
DefaultImageFont	arial	; Standardfont
DefaultImageSize	14	; Schrift in Punkt
FontPath	css/	; Fontverzeichnis

[function defaults]

V

S

I

1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E

Im Fontverzeichnis müssen die TTF-Dateien der verwendeten Fonts liegen. Die Grafikfunktionen arbeiten nur mit TTF-Fonts. Im Internet finden Sie viele Quellen für interessante, freie TTF-Fonts.

11.4 Grundlegende Hinweise zum Prinzip

Dieser Abschnitt enthält Informationen über die prinzipielle Arbeitsweise. Dadurch wird das Verständnis für die Arbeitsweise erleichtert.

11.4.1 Prinzip der Linkauflösung

phpTemple arbeitet mit XML-Tags, die in die HTML-Vorlagen eingebettet werden und bestimmte Funktionen aufrufen, die ihrerseits in PHP ausgeführt werden. Damit die komplette Site dynamisch und ohne Programmierung verwaltet werden kann, verwendet *phpTemple* eine sogenannte Linkauflösung. Die Linkauflösung basiert auf einer Tabelle, `xx_linkresolve`, die Link-Namen zu Vorlagen zuordnet. Die Dateien werden unterhalb des Vorlagenstammverzeichnisses platziert. Sie können dort also eine beliebige Struktur aufbauen. Auch mehrere Vorlagen mit gleichen Namen sind kein Problem. Allein durch die Ablage in verschiedenen Verzeichnissen und der Verknüpfung in der Tabelle kann der Konflikt gelöst werden. Damit werden auch Projekte, an denen mehrere unabhängige Webdesigner arbeiten, leicht beherrschbar.

Die Linkauflösungstabelle

Die Linkauflösungstabelle hat folgende Struktur:

```
CREATE TABLE xx_linkresolve (
  lrsv_id bigint(20) NOT NULL auto_increment,
  lrsv_name varchar(50) NOT NULL,
  lrsv_dir varchar(30),
  lrsv_file varchar(80) NOT NULL,
  lrsv_desc varchar(255),
  PRIMARY KEY (lrsv_id),
  UNIQUE lrsv_name_2 (lrsv_name),
  KEY lrsv_name (lrsv_name)
);
```

Dabei wird bei der Installation der Präfix »xx« durch den gewählten Namensraum ersetzt. Sie können also in einer Datenbank problemlos mehrere völlig voneinander unabhängige Installationen starten.

Feldinformationen

Die Felder haben folgende Bedeutung:

- *lrsv_name*

Name der Vorlage, wie Sie in der Link-Variable `${LINK_name}` geschrieben wird.

- *lrsv_dir*

Verzeichnis unterhalb des Stammverzeichnisses der Vorlagen. Die Angabe eines führenden Schrägstriches ist nicht notwendig.

- *lrsv_file*

Dateiname, der die Vorlage enthält einschließlich Erweiterung.

- *lrsv_desc*

Optionales Feld mit einer Beschreibung für die Administration. Auf die Ausführung hat dies keinen Einfluss.

11.4.2 Einschränkungen

phpTemple verwendet intern zur Verarbeitung der Vorlagen ein Array. Das ist schneller als die Verarbeitung von Zeichenketten, vor allem, wenn viel mit Schleifen gearbeitet wird und immer wieder eine bestimmte Position aufgesucht werden muss. Daraus ergeben sich Einschränkungen beim Aufbau der Vorlagen. Dies verhindert keine Funktion, sondern ist nur ein Merkmal, dass Webdesignern mitgeteilt werden muss.

Jeder Befehl, der von *phpTemple* verarbeitet werden soll, muss allein auf einer Zeile stehen. Leerzeichen oder Tabulatoren davor oder danach stören nicht, bei allen anderen Zeichen versagt die Erkennung.

Eine Ausnahme sind Variablen, die überall eingesetzt werden können, auch mehrfach auf einer Zeile und in den Parametern der Tags.

11.5 Anwendungsbeispiele

Dieser Abschnitt zeigt einige kleine Vorlagen, die typische Lösungen für alltägliche Probleme enthalten.

11.5.1 Eine Navigation aufbauen

Eine Navigation sollte robust gegen Änderungen sein. Es ist sinnvoll, diese an zentraler Stelle einzubauen – nur eine Vorlage sollte die Navigation enthalten.

Angenommen, Sie haben vier Funktionen:

V

S

I

1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E

- Suchen
- Anmelden
- Kontakt
- Hilfe

Realisieren Sie eine Teilvorlage dafür, die Sie folgendermaßen einbinden:

```
<LS:INCLUDE TEMPLATE="navigation"/>
```

Die Teilvorlage verwendet Bilder, die folgendermaßen selektiert werden:

```
<LS_IF VAR1="{navigation}" OP="==" VAR2="nav1">
  
</LS_IF>
<LS_IF VAR1="{navigation}" OP="==" VAR2="nav2">
  
</LS_IF>
<LS_IF VAR1="{navigation}" OP="==" VAR2="nav3">
  
</LS_IF>
<LS_IF VAR1="{navigation}" OP="==" VAR2="nav4">
  
</LS_IF>
```

In den eigentlichen Vorlagen schreiben Sie dann:

```
<LS:SET VAR="navigation" VALUE="nav1"/>
```

Wenn nun eine solche Vorlage aufgerufen wird, müssen Sie nur darauf achten, dass die Definition am Anfang erfolgt, also bevor die Teilvorlage eingebunden wird:

```
<html>
<head>
<LS:SET VAR="navigation" VALUE="nav1"/>
</head>
<body>
<LS:INCLUDE TEMPLATE="navigation"/>
... weiter auf der Seite
</body>
</html>
```

So wird immer das passende Bild in der Navigation angezeigt. Änderungen sind sehr einfach an zentraler Stelle vorzunehmen.

11.5.2 Grafiken dynamisch erzeugen

Das Erzeugen dynamischer Seiten ist nicht einfach. Eigentlich ist es eine Aufgabe des Webdesigners. Praktisch bleibt dieser Teil bei der

Programmierung liegen. Mit *phpTemple* ist es sehr einfach, Grafiken zu erzeugen.

Balkendiagramme erstellen

Die folgende Vorlage liest eine Datenbank aus und stellt die Werte grafisch in Form eines Balkendiagramms dar.

```
<LS:DATA LOOPS="0" STMT="SELECT value, value * 10 AS barvalue  
FROM ls_graph">  
  <LS:IMG WIDTH="{barvalue}" HEIGHT="25" TEXT="{value}"  
    FONTFACE="verdana" FONTSIZE="12" COLOR="FFFFFF"  
    BGCOLOR="666600" ALIGN="left" BORDER="1"/> <br>  
</LS:DATA>
```

Die Ausgabe lässt sich mit den gezeigten Parametern leicht modifizieren.

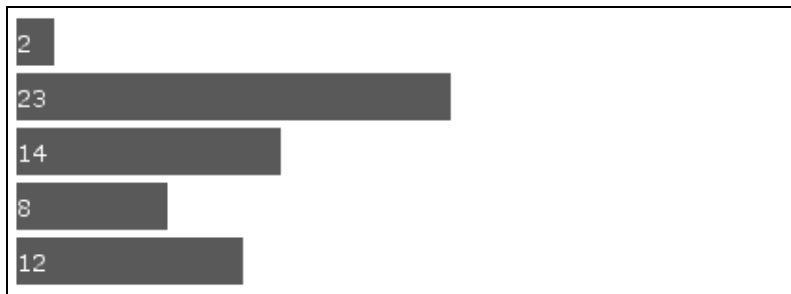


Abbildung 11.3:
Ausgabe des
Balkengrafikanzeige

Wenn Sie den Parameter `ROTATE="90"` hinzufügen, wird die Schrift in der Grafik gekippt. Vertauschen Sie noch `WIDTH` mit `HEIGHT`, stehen die Balken aufrecht, wie in der folgenden Abbildung gezeigt:

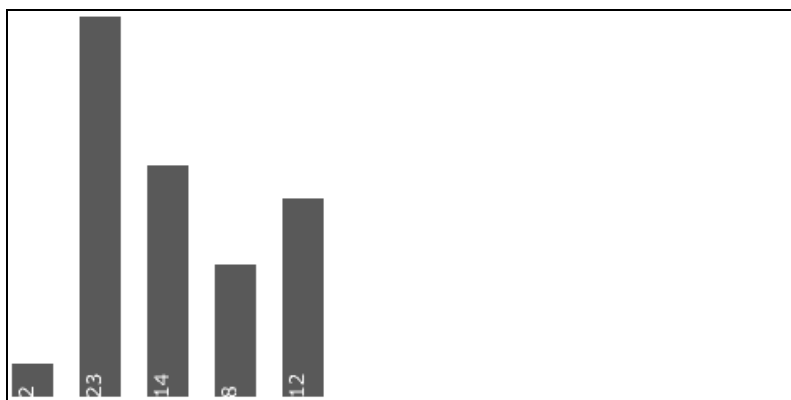


Abbildung 11.4:
Ausgabe vertikaler
Balken mit
Beschriftung

Eine Mischung der Schriftrichtungen ist nicht möglich. Sie können dies aber leicht simulieren, indem zwei Grafiken unmittelbar nebeneinander gesetzt werden.

11.6 Realisierung: So funktioniert es!

In diesem Abschnitt werden die wichtigsten Passagen der Skripte, aus denen *phpTemple* besteht, ausführlich kommentiert vorgestellt. Die abgedruckten Codes basieren auf der OpenSource-Version 0.9, die im Sommer 2001 fertiggestellt wurde.

11.6.1 Startdatei

Die Startdatei INDEX.PHP4 führt jedes Skript aus. Praktisch gibt es deshalb nur Aufrufe dieser Datei. Durch Parameter wird gesteuert, welche Vorlage verwendet werden soll. Einzig notwendig ist der Parameter TEMPLATE:

```
http://<server>.<domain>/<site>/index.php4?TEMPLATE=name
```

Innerhalb von INDEX.PHP4 wird dann die Linkauflösung vorgenommen und der Name durch den Pfad der Vorlagendatei ersetzt. In der Praxis ist dies nicht sichtbar, weil die URL verschlüsselt wird.

Aufbau der Startdatei

Die Startdatei erledigt außer der Linkauflösung vor allem das Einschließen der Moduldateien.

Zuerst wird die globale Zeit festgelegt.

```
<?php
$GTIME = time();
```

Anschließend wird die Konfigurationsdatei geladen. Hier werden auch die Datenbankparameter festgelegt. Im Wesentlichen stehen in der Datei CONFIG.INC.PHP4 die aus der LS.INI ausgelesenen Werte als Konstanten-Definitionen.

```
if (file_exists("config.inc.php4"))
{
    include_once('config.inc.php4');
} else {
    die("<b>Temple</b> was not installed correctly. Please edit
        `ls.ini` and rerun `install.php4` from root dir");
}
```

Als erstes Modul wird das Fehlermodul eingebunden:

```
require_once('errors.inc' . LS_PX);
$objError = new LS_err;
```

Es folgen die Datenbankfunktionen und die Instanziierung des Datenbankobjekts:

```
include('adodb.inc.php4');
$objDB = &ADONewConnection(LS_DBTYPE);
```

```
$objDB->PConnect(LS_DBSERVER.':'.LS_DBPORT, LS_DBUSERNAME,
                LS_DBPASSWORD, LS_DBDATABASE);
```

Nun wird der Parser eingebunden:

```
include_once('parser.inc' . LS_PX);
$objSite = new LS_parser;
```

Es folgt mit der Linkauflösung die einzige globale Funktion in *phpTemplate*:

```
function linkresolve($templatename = '')
{
    global $objDB, $objError;
    if ($templatename != '')
    {
        $objRS = &$objDB->Execute("SELECT * FROM ".LS_TABLE_RESOLVE
                                . " WHERE lrsv_name =
                                '$templatename'");
        if ($objRS->RowCount() == 1)
        {
            return sprintf('%s/%s', $objRS->Fields('lrsv_dir'),
                           $objRS->Fields('lrsv_file'));
        } else {
            $arrErr['source'] = $templatename;
            trigger_error($objError->gen_msg(LSERR_TMPNOTFOUND,
                                             $arrErr), E_USER_WARNING);
            $template = FALSE;
            return FALSE;
        } /* end if */
    } /* end if */
} /* end method linkresolve */
```

Dann werden einige interne Variablen als Referenz für andere Bereiche zur Verfügung gestellt, vor allem für Servlets.

```
$TO = &$objSite;
$SV = &$objSite->arrSiteVariables;
$SS = &$objSite->arrSessionVariables;
$PV = &$objSite->arrProcessVariables;
```

Es folgt die Einbindung des Session-Moduls:

```
include_once('session.inc' . LS_PX);
$objSess = new LS_session;
```

Das nächste Modul enthält die Formularbehandlung:

```
include_once('formval.inc' . LS_PX);
$objForm = new LS_formvalidator;
```

Dann kommt die Definition eines Stammverzeichnisses. Über diesen Parameter wird die Administration gesteuert.

```
define('LS_BaseDir', LS_TemplateDir);
```

Im letzten Schritt kann die Vorlage nun verarbeitet werden. Falls die Linkauflösung misslang, wird ein Fehler angezeigt. Wurde jedoch überhaupt kein Template angegeben, wird die Stammvorlage (Konstante `LS_Home`) aufgerufen.

```
$objSite->Read_Template(  
    LS_TemplateDir . ((strlen($SV['TEMPLATE']) == 0) ?  
    LS_Home : linkresolve($SV['TEMPLATE']));  
$objSite->Parse_Template();  
?>
```

11.6.2 Sessionverwaltung

Die Sessionverwaltung von PHP wird fast vollständig durch eine eigene ersetzt. Diese basiert auf der Datenbank – ohne Datenbank sind keine Sessions möglich. Dafür werden alle Werte sicher in einer Tabelle gespeichert, was mehrere Vorteile hat:

- Keine Manipulation von außen möglich
- Keine Cookies erforderlich
- Abruf der Anzahl der aktiven Sessions einfach möglich

Sessiontabelle

Die Tabelle für die Sessionverwaltung hat folgenden Aufbau:

```
CREATE TABLE xx_session (  
    sessionID varchar(32) NOT NULL,  
    usrID bigint(20) DEFAULT '0' NOT NULL,  
    variables text NOT NULL,  
    laccess int(14),  
    PRIMARY KEY (sessionID),  
    KEY usrID (usrID)  
);
```

Die Session-ID, die von PHP erzeugt wird, dient als Primärschlüssel. Da *phpTemple* leicht um eine Benutzerverwaltung ergänzt werden kann, ist dafür das Feld schon vorbereitet, `usrID` verknüpft die Benutzernummer mit einer laufenden Session. Die Sessionvariablen werden im Feld `variables` abgelegt. Für die Speicherbereinigungsroutine und zum Messen der Sitzungszeit wird der Zeitstempel des letzten Zugriffs in `laccess` gespeichert.

Code des Sessionskripts

Nachfolgend wird das Sessionskript `SESSION.INC.PHP4` vorgestellt. Der Abdruck verzichtet jedoch auf Kommentare und Allgemeinplätze. Das vollständige Skript finden Sie auf der CD und im Web.

Die komplette Sessionverwaltung ist in der Klasse *LS_session* verpackt. Damit werden Namenskonflikte mit anderen Programmteilen vermieden und die Anwendung vereinfacht sich. **Erläuterung der Funktionsweise**

```
class LS_session
{
```

Es folgen mehrere Variablendeklarationen. Dann wird der Constructor der Klasse definiert.

```
function LS_session ()
{
```

Zuerst wird das zentrale Parserobjekt *\$objSite* und das Datenbankobjekt *\$objDB* für die Klasse verfügbar gemacht:

```
$this->objSessDB = $GLOBALS['objDB'];
$this->objSite = $GLOBALS['objSite'];
```

Zusätzlich erfolgt der Zugriff auf den globalen Zeitstempel. Damit steht allen Skriptteilen eine einheitliche Zeit zur Verfügung, auch wenn auf langsamen Servern zwischen den Funktionsaufrufen mehrere Sekunden vergehen sollten.

```
$this->GTIME = $GLOBALS['GTIME'];
```

Nun wird die interne Routine der Sitzungsverwaltung deaktiviert und die eigenen Funktionen – Methoden der aktuellen Klasse – werden definiert:

```
session_module_name('user');
session_set_save_handler(array ($this, 'lsess_open'),
    '',
    array ($this, 'lsess_read'),
    array ($this, 'lsess_write'),
    array ($this, 'lsess_destroy'),
    array ($this, 'lsess_gc'));
```

Der Name der Sitzung wird auf die Kodierung des Projekts festgelegt:

```
session_name(LS_NS);
```

Dann wird die automatische Ersetzung der Links abgeschaltet und Cookies werden unterdrückt. Die Unterdrückung ist notwendig, weil die Links verschlüsselt werden.

```
ini_set('session.use_trans_sid', 0);
ini_set('session.use_cookies', 0);
```

Da der Constructor die erste Funktion ist, die nach dem Seitenstart aufgerufen wird, werden nun alle Variablen der verschlüsselten URLs ermittelt. Der QueryString kann folgenden Aufbau haben:

```
xx=HAHFDJW834hDREWdsa348Sjse2348&uncodiert=wert
```

xx steht für den Namensraum der Installation. Unkodierte Teile entstehen, wenn Parameter clientseitig, beispielsweise durch JavaScript angehängt werden.

```
foreach($GLOBALS['HTTP_GET_VARS'] as $strParm => $strValue)
{
    if ($strParm == LS_NS)
    {
        $arrURI = unserialize(base64_decode($strValue));
        foreach(explode('&', $arrURI['query']) as $strQParm)
        {
            list($strVarName, $varVarValue) = explode('=', $strQParm);
            $this->objSite->arrSiteVariables[$strVarName]
                = $varVarValue;
        }
    } else {
        $this->objSite->arrSiteVariables[$strParm] = $strValue;
    }
}
```

Standardmäßig werden zwei Parameter verschlüsselt verwendet: query enthält den ursprünglichen QueryString, sid die Session-ID. Wenn Formulare verwendet werden, erzeugt *phpTemple* ein verstecktes Feld mit der Session-ID. Der folgende Teil ermittelt, wie die Session-ID übertragen wurde. Der Wert wird dann zur gültigen Session-ID erklärt (Funktion `session_id`):

```
if (isset($arrURI['sid']))
{
    session_id($arrURI['sid']);
} else if (isset($GLOBALS['HTTP_POST_VARS']['sid']))
{
    session_id($GLOBALS['HTTP_POST_VARS']['sid']);
}
```

Nun kann die Session starten. Wurde keine ID übermittelt, erzeugt PHP selbst eine:

```
session_start();
```

Die Session-ID wird für die Vorlagen als Variable bereitgestellt, ebenso die Informationen über den angemeldeten Benutzer:

```
$this->objSite->arrSiteVariables['SESSIONID']
    = $this->sessionID = session_id();
$strQuery = "SELECT u.usrid, u.logname
FROM " . LS_TABLE_SESSION . " cs, usr u
WHERE cs.sessionID = '$this->sessionID'
AND cs.usrID = u.usrID";
$objRS = &$this->objSessDB->Execute($strQuery);
$this->objSite->arrSiteVariables['USERID'] = $objRS->Fields[0];
$this->objSite->arrSiteVariables['USERNAME'] = $objRS->Fields[1];
```

In jeder Vorlage stehen nun die Variablen `${SESSIONID}`, `${USERID}` und `${USERNAME}` zur Verfügung.

Es folgen einige weitere feste Werte, wie die aktuellen Zeit- und Datumsangaben in der Standardsprache:

```
$strOldLangCode = setlocale(LC_TIME, NULL);
setlocale(LC_TIME, LS_LANG);
$this->objSite->arrSiteVariables['LONGDATE'] = strftime("%A,%x");
$this->objSite->arrSiteVariables['SHORTDATE'] = strftime("%x");
$this->objSite->arrSiteVariables['LONGTIME'] = strftime("%X");
$this->objSite->arrSiteVariables['SHORTTIME'] = strftime("%H:%M");
setlocale(LS_TIME, $strOldLangCode);
$this->objSite->arrSiteVariables['LANGUAGE'] = LS_LANG;
```

Zuletzt wird noch verhindert, dass die laufende Session abbricht und ein definiertes Ende der Session eingerichtet:

```
ignore_user_abort();
register_shutdown_function(array ($this, 'end_session'));
```

Damit ist der Constructor beendet. Es folgt die Methode, die am Anfang jeder Seite als erstes zur Eröffnung der laufenden Session aufgerufen wird:

```
function lsess_open($strPath, $strName)
{
```

Zuerst werden die Daten der aktuellen Session abgerufen.

```
$strQuery = "SELECT * FROM " . LS_TABLE_SESSION . "
            WHERE sessionID = '$this->sessionID'";
$objRS = &$this->objSessDB->Execute($strQuery);
```

Wenn für diese Session-ID noch kein Eintrag existiert, wird ein neuer Eintrag erzeugt.

```
if ($objRS === FALSE or $objRS->RecordCount() == 0)
{
    $strQuery = "INSERT INTO " . LS_TABLE_SESSION .
                " (sessionID, laccess)
                VALUES ('$this->sessionID', '$GTIME')";
```

Andernfalls wird lediglich der Zeitstempel aktualisiert, da die Session noch verwendet wird.

```
    } else {
        $strQuery = "UPDATE " . LS_TABLE_SESSION . "
                    SET laccess = '$GTIME'
                    WHERE sessionID = '$this->sessionID'";
    }
    $objRS = $this->objSessDB->Execute($strQuery);
    return $bInRS;
} /* end function */
```

Nach dem Eröffnen der Session auf einer Seite wird automatisch die Methode `lsess_read` aufgerufen; dies erledigt PHP, einen direkten Aufruf gibt es nicht.

```
function lsess_read($strSessID)
{
```

Hier wird auch der Abruf der globalen Variablen durchgeführt, die damit in jedem Skript unabhängig von der Session zur Verfügung stehen:

```
    $strQuery = "SELECT variable, value FROM " . LS_TABLE_GLOBALS;
    $objRS = &$this->objSessDB->Execute($strQuery);
    while (!$objRS->EOF)
    {
        $this->objSite->arrSiteVariables[$objRS->Fields('variable')]
            = $objRS->Fields('value');
        $objRS->MoveNext();
    }
```

Dann werden die Variablen der aktuellen Session gelesen:

```
    $strQuery = "SELECT variables FROM " . LS_TABLE_SESSION . "
        WHERE sessionID = '$strSessID'";
    $objRS = $this->objSessDB->Execute($strQuery);
    if ($objRS->RecordCount() == 1)
    {
```

Die Umwandlung der serialisierten Werte erledigt PHP wieder intern, wenn die Zeichenkette zurückgegeben wird. Ebenso wie beim Aufruf der Funktion gibt es kein sichtbares Ziel dieser Rückgabe.

```
        return $objRS->Fields('variables');
    } else {
        return FALSE;
    }
} /* end method */
```

Die nächste Methode erledigt am Ende einer Seite das Schreiben der Sessionvariablen in die Datenbank. Auf diese Methode ruft PHP automatisch auf und übergibt die serialisierten Werte im zweiten Parameter `$strVars`.

```
function lsess_write($strSessID, $strVars)
{
    $strQuery = "UPDATE " . LS_TABLE_SESSION . "
        SET variables = '$strVars'
        WHERE sessionID = '$strSessID'";
    $this->objSessDB->Execute($strQuery);
```

Für alle Fälle werden außerdem die Standardwerte aus der `PHP.INI` wieder restauriert.

```

ini_restore('session.use_trans_sid');
ini_restore('session.use_cookies');
return TRUE;
}

```

Falls die Session abbricht oder gezielt gelöscht wird (diese Funktion wird jedoch nicht intern verwendet), erledigt dies die folgende Methode.

```

function lsess_destroy($strSessID)
{
    $strQuery = "DELETE FROM " . LS_TABLE_SESSION . "
                WHERE sessionID = '$strSessID'";
    $bInRS = $this->objSessDB->query($strQuery);
    return $bInRS;
}

```

Ebenso wie die anderen Methoden wird auch die Speicheraufräumroutine automatisch gestartet. PHP entscheidet anhand eines Verhältnswertes, wann dieser Prozess läuft. Zusätzlich zum Löschen der alten Session werden auch an Sessions gebundene gecachte Dateien im Cache entfernt.

```

function lsess_gc($tSesslt)
{
    $tStamp = $this->GTIME - $tSesslt;

```

Zuerst werden alle Sessions ermittelt, die bereits zu alt sind:

```

    $strQuery = "SELECT sessionID FROM " . LS_TABLE_SESSION . "
                WHERE laccess < '$tStamp'";
    $objRS = $this->objSessDB->Execute($strQuery);

```

Daraus wird ein Array der Session-IDs erstellt.

```

    while (!$objRS->EOF)
    {
        $arrSessionIDs[] = $objRS->Fields('sessionID');
    }

```

Dann wird eine Liste der unter dem Namen der Session gespeicherten, sessiongebundenen Cachedateien erstellt:

```

    $hdlD = opendir(_CacheDir);
    while ($strFile = readdir($hdlD))
    {
        $arrFile = explode('@', $strFile);
        $arrFileIDs[] = $arrFile[1];
        $arrFiles[] = $strFile;
    }
    if (is_array($arrSessionIDs) and is_array($arrFileIDs))
    {

```

Aus den beiden Arrays wird die Schnittmenge gebildet, um die zu löschenden Dateien zu ermitteln:


```
$arrDeleteFiles=array_intersect($arrSessionIDs, $arrFileIDs);
```

Diese Dateien werden dann gelöscht:

```
for($i = 0; $i < count($arrDeleteFiles); $i++)
{
    unlink (_CacheDir . '/' . $arrFiles[$i]);
}
```

Im letzten Schritt werden die veralteten Sessions in der Sessiontabelle entfernt.

```
$strQuery = "DELETE FROM " . LS_TABLE_SESSION . "
            WHERE laccess < '$tStamp'";
$objjRS = $this->objSessDB->Execute($strQuery);
return $objjRS;
}
```

Falls eine Session aufgrund eines Verbindungsabbruchs endet, wird sie gezielt zerstört, was die folgende Funktion erledigt:

```
function end_session()
{
    if (connection_aborted()) $this->l sess_destroy();
}
```

Vorteile

Insgesamt ist diese Sessionverwaltung extrem sicher gegenüber Angriffen von Hackern, robust und über die Datenbank sehr gut skalierbar. Da praktisch (außer den Cache-Dateien) keine lokal gespeicherten Daten vorhanden sind, kann PHP im Rahmen einer Lastverteilung (Load Balancing) auf mehrere Computer verteilt werden. Nur die Datenbank läuft auf einer größeren Maschine, beispielsweise mit Oracle oder MS SQL Server. Da alle Datenbankzugriffe in *phpTemple* datenbankunabhängig sind, ist ein Austausch kein Problem und verlangt keine Änderungen am Code.

11.6.3 Das Kernstück: der Parser

Der gesamte Parser besteht aus über 1 300 Zeilen Code. Es ist unmöglich, diese am Stück hier abzudrucken. Deshalb wird das Prinzip am Beispiel einiger einfacher Befehle erläutert. Auch dieses Modul ist als Klasse verpackt.

Funktionsweise

Der Parser arbeitet relativ einfach – auch wenn die letztlich aktive Codemenge erheblich ist. Am Anfang wird die komplette Vorlage in ein Array *arrCurrentSite* eingelesen. Dieses Array wird dann zeilenweise verarbeitet. Jede Zeile wird mit Hilfe eines regulären Ausdrucks untersucht. Findet der Parser ein Tag, wird dieses weiter analysiert

und die entsprechenden Funktionen werden ausgeführt. Daten, die nicht verarbeitet werden müssen, werden unverändert in das Ergebnisarray *arrOutputBuffer* übertragen, ebenso wie die Ergebnisse der Funktionen. Am Ende werden einige globale Aktionen ausgeführt, nachdem das Ergebnisarray wieder zu einer Zeichenkette zusammengeführt wurde. Mit Hilfe der Bufferfunktionen von PHP werden die letzten Änderungen im Buffer ausgeführt und die fertige Datei dann mit einem Mal übertragen.

Der Parser im Detail

```
class LS_parser {
```

Am Anfang der Definition stehen die Deklarationen für die Eigenschaften. Dann folgt der Constructor; diese Methode stellt die Anfangszustände her. Drei globale Variablen werden außerdem verwendet: *\$objDB* greift auf die Datenbank zu, *\$GTIME* stellt die globale Zeit zur Verfügung und *\$objError* ist eine Instanz der Fehlerklasse. Aufgrund der Komplexität von *phpTemple* bot sich eine eigene Fehlerverwaltung an.

```
function LS_parser() {  
    global $objDB;  
    global $GTIME;  
    global $objError;  
    ...  
}
```

Die weiteren Definitionen sollen hier nicht betrachtet werden. Einzig von Bedeutung könnte die Initialisierung der HTML-Kodierfunktion sein, mit der Umlaute in HTML-Entitäten gewandelt werden:

```
$this->arrEncoding = array("ö" => "&ouml;", "ä" => "&auml;",  
                           "ü" => "&uuml;", "õ" => "&Ouml;",  
                           "Ä" => "&Auml;", "Ü" => "&Uuml;",  
                           "ß" => "&szlig;");
```

Es folgt eine ganze Palette von Hilfsfunktionen, die während des Verarbeitungsprozesses benötigt werden.

Zuerst eine Methode, die die innerhalb eines <LS:DATA>-Blocks definierten Variablen wieder löscht.

Hilfsfunktionen

```
function /* private */ Clear_SQLBlockVariables()  
{  
    ...  
}
```

Dann folgt die Methode, die aus dem Ergebnissatz einer Datenbankabfrage die Variablen für die Seite ersetzt.

```
function /* private */ Set_SQLBlockVariables($arrResultArray)  
{
```

```
...
}
```

An vielen Stellen, in Attributen und auch einfach im Text, werden Variablen auftreten. Wird eine Variable erkannt, sorgt die Methode *Resolve_Variables* dafür, dass der Inhalt der Variable statt des Namens eingesetzt wird.

```
function /* private */ Resolve_Variables($strVarName)
{
    if (isset($this->arrSiteVariables[$strVarName]))
    {
        $varReplaceValue = $this->arrSiteVariables[$strVarName];
    } else {
        $varReplaceValue = '';
    } /* end if */
    return $varReplaceValue;
} /* end function Resolve_variables */
```

Die nächste Methode registriert Sessionvariablen. Hier wird mit einem kleinen Trick gearbeitet, weil PHP Variablen nur dann registriert, wenn diese global sind. Deshalb wird der Funktion ein Name und ein Wert übergeben und daraus eine globale Variable erstellt. Dieser wird dann über die Namensreferenz (*\$\$strName*) ein Wert zugewiesen und PHP sorgt dann mit *session_register* für die Registrierung und Serialisierung.

```
function /* private */ register_sessionvar($strName,
                                           $strValue = '', $bInDo = TRUE)
{
    if ($bInDo)
    {
        global $$strName;
        $$strName = $strValue;
        session_register($strName);
        return TRUE;
    } else {
        return FALSE;
    }
}
```

Prüfmethoden für Tags

Es folgen nun einige Prüfmethoden für Tags, die beim parsen der Zeilen aufgerufen werden. Die erste Methode prüft, ob ein Tag geschlossen wurde. Alleinstehende Tags im XML-Stil müssen geschlossen werden. Falls der schließende Schrägstrich fehlt, wird ein Fehler erzeugt. *trigger_error* ruft die benutzerdefinierte Fehlerverwaltung auf. Dies wird im Abschnitt XXX näher erläutert.

```
function /* private */ check_closed($strTag)
{
    if (!$this->bInClosed)
    {
```

```

        $arrErr = array('file' => $this->strCurrentTemplate,
                        'line' => $this->intLineCounter,
                        'tag' => $strTag);
        trigger_error($this->objError->gen_msg(LSERR_TAGNOTCLOSED,
        $arrErr), E_USER_WARNING);
    }
}

```

Die folgende Funktion erzeugt standardisiert Fehlermeldungen, die sehr oft benötigt werden. Letztlich interessiert den Entwickler von Vorlagen ja nicht die Zeile, wo der Fehler im Parser auftrat, sondern die aus dem aktuellen Template (die in *intLineCounter* steht).

```

function /* private */ ErrorData()
{
    return array('file' => $this->strCurrentTemplate,
                'line' => $this->intLineCounter,
                'tag' => $this->strTag);
}

```

Manchmal kommen Sie nicht umhin, komplexe Prozesse doch in PHP ausführen zu lassen. Die folgende Methode sorgt für die Ausführung. Die Kapselung in einer Funktion trennt die Namensräume konsequent, sodass keine Variablen des Parsers versehentlich überschrieben werden können. Die drei Variablen *\$DB*, *\$SV* und *\$PV* reichen die entsprechenden Parsevariablen an das Skript durch und vereinfachen lediglich den Aufruf. Diese Verfahrensweise ist zwar primitiv, schützt aber vor Überschreiben (der Zugriff ist quasi »read only«), kapselt aber nicht so stark wie explizite Funktionsaufrufe. Dafür ist dies schneller, was bei in Hochsprachen geschriebenen Parsern sehr wichtig ist.

```

function /* private */ Process_Servlet($strServlet)
{
    $DB = $this->objDBLink;
    $SV = $this->arrSiteVariables;
    $PV = $this->arrProcessVariables;
    require(LS_ServletDir . $strServlet . LS_SE);
    return $SV['return'];
}

```

Die nächste Methode löst Variablen in Parametern auf. Dabei werden die besonderen Systemvariablen erkannt, die immer nach dem Schema {PRÄFIX_name} aufgebaut sind.

```

function /* private */ Resolve_Parameter($strParameter)
{

```

Zuerst wird ermittelt, ob überhaupt eine Variable enthalten ist.

```

    if (preg_match_all(LS_REGEX_TRVAR, $strParameter, $arrContent))
    {

```

Dann werden alle Vorkommen nacheinander aufgelöst:

```
foreach($arrContent[1] as $strVarName)
{
```

Um den Präfix abzutrennen, wird `split` verwendet. Im Gegensatz zu `explode` kann hier die Zerlegung auf den ersten Unterstrich im Namen beschränkt werden. Damit unterliegen die eigenen Namensteile, die ja von der Programmierung der Vorlagen abhängen, keinen Einschränkungen.

```
$arrSwitch = split('_', $strVarName, 2);
switch ($arrSwitch[0])
{
```

Felder beginnen mit `FIELD`. Diese werden, ebenso wie die anderen Feldwerte, einfach durch den Inhalt ersetzt. Das »Befüllen« der Feldvariablen geschieht übrigens im Modul `FORMVAL.INC.PHP4`.

```
case 'FIELD': case 'ERROR':
case 'CHECKED' : case 'SELECTED':
    $varReplaceValue
        = addslashes($this->arrSiteVariables
            ["$arrSwitch[0]_{$arrSwitch[1]}"]);
    break;
```

Links werden mit `LINK` eingeleitet und durch den Aufruf der Stammdatei (standardmäßig `INDEX.PHP4`) und den Parameter `TEMPLATE` übertragen.

```
case 'LINK':
    $varReplaceValue = LS_MAIN.'?TEMPLATE='.$arrSwitch[1];
    break;
```

Formulare werden immer an die Stammdatei gesendet, `#{MAIN}` ist deshalb vor allem in `<form>`-Tags zu finden.

```
case 'MAIN':
    $varReplaceValue = LS_MAIN;
    break;
```

Navigationssteuerung

Etwas aufwändiger ist die Navigationssteuerung. Die Navigation basiert auf mehreren vordefinierten Variablen (beispielsweise enthält `NAVI_NEXT` die Nummer der nächsten Seite einer Liste). Der folgende Block erkennt diese Variablen und ersetzt sie durch die berechneten Werte. Die Daten, die Auskunft über die aktuelle Position in der Navigation geben, stehen in den Variablen mit dem Präfix `INFO`.

```
case 'NAVI':
    $intOffset = $this->arrSiteVariables['INFO_OFFSET'];
    $intPageSize=$this->arrSiteVariables['INFO_PAGESIZE'];
    $intTotal = $this->arrSiteVariables['INFO_SUM'];
    $intMaxPage = ceil($intTotal / $intPageSize) - 1;
```

```
switch ($arrSwitch[1])
{
```

FIRST wird immer auf 0 gesetzt (die Seitenzählung beginnt bei 0):

```
case 'FIRST':
    $varReplaceValue = '0';
    break;
```

PREVIOUS wird um eins verringert, wenn der ursprüngliche Wert noch größer als 0 ist, andernfalls bleibt der Wert bei 0 stehen. Damit wird verhindert, dass eine Navigation auf nicht vorhandene Seiten führt. Ebenso wird bei NEXT das Ende der Seitenliste erkannt.

```
case 'PREVIOUS':
    $varReplaceValue = ($intOffset > 1)
        ? $intOffset - 1 : 0;
    break;
case 'NEXT':
    $varReplaceValue = ($intOffset < $intMaxPage)
        ? $intOffset+1 : $intMaxPage;
    break;
case 'LAST':
    $varReplaceValue = $intMaxPage;
    break;
} /* end switch */
break;
```

Die Informationswerte selbst werden nicht nur zur Steuerung der Navigation verwendet, sondern stehen auch dem Programmierer der Vorlage zur Verfügung. Der folgende Block ersetzt sie durch die entsprechenden Werte. Beachten Sie, dass die Berechnung nur erfolgt, wenn diese Variablen auch auftreten. Dies ist ein Grund für die hohe Leistung von *phpTemple*: Nur tatsächlich benötigte Daten führen zu aufwändigen Berechnungen.

```
case 'INFO':
    $intOffset
        = (int) $this->arrSiteVariables['INFO_OFFSET'];
    $intPageSize
        = (int) $this->arrSiteVariables['INFO_PAGESIZE'];
    $intTotal
        = (int) $this->arrSiteVariables['INFO_SUM'];
```

Mit der Funktion `ceil` wird die Anzahl der Seiten berechnet:

```
$intMaxPage = ceil($intTotal / $intPageSize);
```

Das erste Element der aktuellen Seite wird folgendermaßen ermittelt:

```
$intFirstElement = $intOffset * $intPageSize + 1;
```

Mit Hilfe der `min`-Funktion wird das letzte Element der aktuellen Seite berechnet:

```
$intLastElement = min($intFirstElement +
    $intPageSize - 1, $intTotal);
```

Dann werden die Werte den Informationsvariablen zugewiesen:

```
switch ($arrSwitch[1])
{
    case 'FROM':
        $varReplaceValue = $intFirstElement;
        break;
    case 'TO':
        $varReplaceValue = $intLastElement;
        break;
    case 'SUM':
        $varReplaceValue = $intTotal;
        break;
```

Alle Variablen, die keine besondere Bedeutung haben, werden einfach durch ihren Inhalt ersetzt. Die Methode *Resolve_Variables* wurde bereits beschrieben:

```
default:
    $varReplaceValue
        = $this->Resolve_Variables($strVarName);
} /* end switch */
$this->arrSiteVariables['INFO_OFFSET'] = $intOffset;
break;
```

Variablen, die bisher nicht erkannt wurden, werden durch den Inhalt ersetzt. Ab diesem Punkt sind alle Variablen erkannt und der Inhalt steht in *\$varReplaceValue*.

```
default:
    $varReplaceValue=$this->Resolve_Variables($strVarName);
} /* end switch trvar resolve */
```

Mit Hilfe der sehr schnellen Funktion *str_replace* wird die Variable nun ersetzt:

```
$strParameter = str_replace('${'.$strVarName.'}',
    $varReplaceValue, $strParameter);
} /* end foreach */
} /* end if */
return $strParameter;
} /* end function Resolve_Parameter */
```

Der Zeilenparser

Die komplexeste Funktion ist der Zeilenparser. Damit wird jeweils eine Zeile durchsucht. Wurde ein Tag gefunden, wird es ausgeführt. In diesem Teil werden nur einige Funktionen beschrieben, da die gesamte Darstellung den Rahmen dieses Buches sprengen würde. Der

Quellcode ist hinreichend dokumentiert, um die Funktion bis ins Detail verstehen zu können.

```
function Parse_Line($strLineToParse)
{
    $strLineToParse = chop($strLineToParse);
    $strTagReplacement = '';
```

Zuerst werden Feldvariablen erkannt, die serverseitig verarbeitet werden sollen. Um den regulären Ausdruck so einfach wie möglich zu halten (komplizierte Ausdrücke sind zwar eleganter, aber leider auch langsamer), werden nur Felddtags erkannt. Die Untersuchung auf das Attribut `runat="server"` folgt später.

```
    if (preg_match_all(LS_REGEX_FIELD,$strLineToParse,$arrContent))
    {
        $strFormfield = strtoupper($arrContent[1][0]);
```

Feldverarbeitung

Ein zweiter Ausdruck löst die Attribute der Felder auf:

```
        preg_match_all(LS_REGEX_TATTR, $arrContent[2][0],
                        $arrContent);
```

Nur wenn eines der Attribute "runat" heißt, wird weiter verarbeitet:

```
        if (in_array('runat', $arrContent[1]))
        {
            $bInServer = FALSE;
```

In einer Schleife werden nun alle Attribute untersucht:

```
        for ($i = 0; $i < count($arrContent[0]); $i++)
        {
            $strDele = $arrContent[0][$i];
            $strParm = $arrContent[1][$i];
            $strAttr = $arrContent[2][$i];
            switch (strtoupper($strParm))
            {
```

Alle Attribute werden gleichartig verarbeitet: Zuerst wird die Aktion ausgeführt oder der Parameter übernommen, danach wird ein Ersatzwert festgelegt (die serverseitig zu verarbeitenden Attribute wie beispielsweise `required` werden entfernt):

```
                case 'RUNAT':
                    if ($strAttr == 'server') $bInServer = TRUE;
                    $strLineToParse = str_replace($strDele, '',
                                                    $strLineToParse);
                    break;
                case 'REQUIRED':
...
                case 'VALIDATIONMSG':
...
                case 'NAME':
...
            }
```

V

S

I

1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E


```

        case 'VALUE':
...
        case 'LANG':
...
        case 'LIST':
...
        case 'ACCEPT':
...
        case 'MAXFILESIZE':
...
        case 'TARGETDIR':
...

```

Das Attribut type wird weiter untersucht, hier muss der Feldtyp erkannt werden:

```

        case 'TYPE':
            $strFieldType=$this->Resolve_Parameter($strAttr);
            $bInFieldCheckbox=$bInFieldRadio=$bInUpload=FALSE;
            switch (strtoupper($strFieldType))
            {
                case 'CHECKBOX':
                    $bInFieldCheckbox = TRUE;
                    break;
                case 'RADIO':
                    $bInFieldRadio = TRUE;
                    break;
                case 'FILE':
                    $bInUpload = TRUE;
                    break;
            }
            break;
        } /* end switch */
    } /* end for */

```

Nach der Analyse aller Parameter erfolgt die Verarbeitung, jedoch nur, wenn tatsächlich eine serverseitige Verarbeitung gewünscht war.

```

        if ($bInServer)
        {
...

```

Jetzt werden die durch die Attribute definierten Werte in der Session gespeichert. Damit stehen diese Informationen auf der nächsten Seite wieder zur Verfügung:

```

        $this->register_sessionvar("PROOF_$strFieldName",
            $strFieldProof, isset($strFieldProof));
        $this->register_sessionvar("ERROR_$strFieldName",
            $strFieldError, isset($strFieldError));
        $this->register_sessionvar("CHECKED_$strFieldName",
            $strFieldValue, $bInFieldCheckbox);

```

```
$this->register_sessionvar("RADIO_{$strFieldName}",
    $strFieldValue, $bInFieldRadio);
```

Der folgende Block steuert die Wiederholung der Auswahl in <select>-Listen und den Aufbau von Standardlisten aus der Tabelle `xx_lists`:

```
switch (TRUE)
{
```

Innerhalb von <option>-Tags wird die Steuervariable `#{AUTOSELECTED}` durch den jeweils aktiven Feldwert ersetzt. Dieser Wert wird in `FORMVAL.INC.PHP4` mit "checked" gefüllt, wenn das Feld ausgewählt war.

```
    case ($strFormfield == 'OPTION'):
        $strLineToParse =
            str_replace('#{AUTOSELECTED}',
                '#{SELECTED_' .
                $this->strCurrentSelectField .
                '_. $strFieldValue.}',
                $strLineToParse);
        break;
    case ($bInAutoList):
...
        case ($bInUpload):
...
    } /* end switch */
    } /* end if $bInServer */
} /* end if */
```

Der nächste Block umfasst die Verarbeitung der Tags. Zuerst werden die Tags überhaupt erkannt: **Tag-Verarbeitung**

```
if (preg_match_all(LS_REGEX_TTAG, $strLineToParse, $arrContent))
{
```

Jetzt wird ermittelt, ob das Tag ein schließendes ist (Schrägstrich am Anfang, wie beispielsweise </LS:IF>):

```
$this->bInClosing = (boolean) $arrContent[1][0];
```

Das Tag selbst wird intern immer zu Großbuchstaben gewandelt, damit spielt Groß- und Kleinschreibung keine Rolle mehr.

```
$strTag = strtoupper($arrContent[2][0]);
```

Die Attribute werden gesondert verarbeitet.

```
$strAttributes = $arrContent[3][0];
```

Dann wird ermittelt, ob das Tag geschlossen wurde (Schrägstrich am Ende, wie beispielsweise <LS:ELSE/>):

```
$this->bInClosed = (boolean) strstr($arrContent[4][0], '/');
```

V

S

I

1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E

Wurden Attribute erkannt, löst diese Strukturen ein weiterer regulärer Ausdruck auf. Die Verarbeitung erfolgt jedoch erst im entsprechenden Block. Damit ist der Programmteil, der »immer« ausgeführt wird, extrem klein. Von dem nachfolgenden switch-Block wird immer nur ein Zweig ausgeführt, sodass tatsächlich nur wenig Code zu verarbeiten ist, was der Systemleistung entgegen kommt.

```
if (strlen($strAttributes) > 0)
{
    preg_match_all(LS_REGEX_TATTR, $strAttributes, $arrContent);
} else {
    $arrContent = array();
}
$this->strTag = $strTag;
```

Der folgende Block verarbeitet die einzelnen Tags.

```
switch ($strTag)
{
```

Zuerst ein einfaches Beispiel. Das Tag <LS:STOP/> enthält keine Attribute.

```
case 'STOP':
```

Verlangt wird, dass dieses Tag immer geschlossen wird. Die Funktion *check_closed* stoppt die Ausführung mit einem Fehler, wenn diese Bedingung nicht erfüllt ist.

```
$this->check_closed($strTag);
```

Dann wird geprüft, ob die Ausführung in einem IF-Block erfolgt und dieser Block aktiv ist. Dies dient der Steuerung des <LS:IF>-Tags.

```
if ($this->intIfActive != IFTRUE) break;
$this->intLineCounter = count($this->arrCurrentSite);
break;
```

Es folgen nun nacheinander alle Tags mit der entsprechenden Verarbeitung:

```
case 'COUNTER':
case 'IMG':
case 'PROCESS':
case 'PERSISTENT':
case 'HEADER':
```

Zur Erläuterung der Arbeitsweise komplexer Tags mag ENCODING dienen. Dieses Tag steuert die Zeichenkodierung (Ä wird zu Ä), und die Kompression.

```
case 'ENCODING':
```

Zuerst folgt die Kontrolle des schließenden Schrägstriches und des IF-Blocks:

```
$this->check_closed($strTag);
if ($this->intIfActive != IFTRUE) break;
```

Die Attribute stehen paarweise in *\$arrContent*. Eine Schleife durchläuft nun alle und speichert die Parameter für die weitere Verarbeitung.

```
for ($i = 0; $i < count($arrContent[0]); $i++)
{
    $strParm = $arrContent[1][$i];
    $strAttr = $arrContent[2][$i];
```

Jedes Attribut wird einzeln ausgewertet. Diese Liste ist einfach zu erweitern, wenn neue Attribute eingeführt werden sollen.

```
switch (strtoupper($strParm))
{
    case 'SWITCH':
```

Die Parameter werden nur gespeichert. In einigen Fällen hängen die Werte voneinander ab, die Reihenfolge soll aber keine Rolle spielen. Deshalb werden die Werte erst zwischengespeichert.

```
        if (strtoupper($strAttr) == 'ON')
            $this->blnEncoding = TRUE;
        if (strtoupper($strAttr) == 'OFF')
            $this->blnEncoding = FALSE;
        break;
    case 'COMPRESS':
        if (strtoupper($strAttr) == 'ON')
            $this->blnCompressing = TRUE;
        if (strtoupper($strAttr) == 'OFF')
            $this->blnCompressing = FALSE;
        break;
```

Der default-Zweig kann nur erreicht werden, wenn unbekannte Attribute auftreten, was hier zum Erzeugen eines Fehlers verwendet wird.

```
        default:
            trigger_error($this->objError->
                gen_msg(LSERR_PARAMUNKNOWN,
                    $this->ErrorData()),
                E_USER_WARNING);
    } /* end switch */
} /* end for */
break;
```

Es folgen die Zweige für weitere Tags:

```
    case 'FOLLOW':
    ...
```

Das Einschließen von Dateien ist insofern interessant, als dass nach der üblichen Analyse der Parameter die Verknüpfung der einzuschließenden Zeilen mit dem bestehenden Array zwei Array-Funktionen nutzt. Zuerst wird mit *array_splice* das aktuelle Array der

gesamten Vorlage zerlegt: *\$this->arrCurrentSite* enthält danach den Teil vom Anfang bis zur aktuellen Position, *\$arrCurrentRest* die letzten Zeilen bis zum Ende.

```

        case 'INCLUDE':
    ...
    if (file_exists(LS_BaseDir . LS_IncludeDir . $strAttr . LS_IE))
    {
        ...
        $arrIncludeSite=file(LS_BaseDir.LS_IncludeDir.$strAttr.LS_IE);
        $arrCurrentRest = array_splice($this->arrCurrentSite,
                                      $this->intLineCounter,
                                      count($this->arrCurrentSite) -
                                      $this->intLineCounter);
        $this->arrCurrentSite = array_merge($this->arrCurrentSite,
                                           $arrIncludeSite,
                                           array('<LS:INCLUDEEND/>'),
                                           $arrCurrentRest);
    }

```

Zwischen dem ersten Teil und dem Rest wird – praktisch am Ende des eingeschlossenen Teils – ein weiteres Tag eingebaut, dass niemals direkt verwendet werden sollte: `<LS:INCLUDEEND/>`. Damit erkennt *phpTemple* später das Ende des eingeschlossenen Bereiches, was für die Steuerung von Fehlermeldungen wichtig ist.

Datenbankzugriffe

Die folgenden Tags verarbeiten Datenbankzugriffe. Dies ist etwas komplexer, weil derartige Abfragen auch verschachtelt werden können und zugleich mit IF-Blöcken steuerbar sind.

```

        case 'DATAWHILE':
        case 'DATA':

```

Die Steuerung der Datenbankabfrage mit `<LS:DATA>`

Die Erläuterung des gesamten Codes für `<LS:DATA>` würde sehr breiten Raum einnehmen. Wenn Sie in den Quellcode schauen, sollten Sie die Funktion einiger Variablen kennen:

- *\$this->intSQLDepth*

Hier wird die Tiefe der Verschachtelung gespeichert.

- *\$this->arrSQLLoopBlock[\$this->intSQLDepth]*

In diesem Array wird die Rücksprungadresse für die Schleifenbildung gespeichert, also die Zeilennummer, zu der gesprungen wird, wenn `</LS:DATA>` den Block beendet und noch weitere Daten vorliegen. Der Index dieses Arrays ist die Ebene aus *\$this->intSQLDepth*, damit verschiedene Verschachtelungstiefen unterschieden werden können.

- `$this->arrSQLLoops[$this->intSQLDepth]`

Hier wird die Anzahl der Schleifendurchläufe gespeichert.

- `$this->strSQLStatement`

Diese Variable enthält die SQL-Anweisung, die verarbeitet werden soll.

- `$this->objRS[$this->intSQLDepth]`

Hier wird der Ergebnisdatensatz der Abfrage gespeichert, der mit jedem Durchlauf der Schleife verwendet wird.

<LS:DATANEXT> und <LS:DATAWHILE> greifen auf diese Variablen ebenfalls zu. Dies setzt jedoch voraus, dass zuerst in <LS:DATA> eine erfolgreiche Datenbankabfrage ausgeführt werden konnte.

Die Steuerung von Blöcken mit <LS:IF>

Wichtig für die Steuerung in der Vorlage sind Verzweigungen. Besonders häufige Standardabfragen sind hier mit speziellen Tags ausgeführt worden, was die Programmierung stark vereinfacht. Da sich diese Tags im Prinzip nur marginal unterscheiden, werden sie zusammen in einem Block ausgeführt.

Auch hier ist der Quelltext gut lesbar, wenn einige Bedingungen bekannt sind:

- `$this->intIfDepth`

Hier wird die Ebene gespeichert, wenn IF-Blöcke verschachtelt sind. Damit wird erkannt, ob ein schließendes IF (</LS:IF>) tatsächlich den aktuellen Block schließt oder zu einer untergeordneten Ebene gehört.

- `$this->intIfActive`

Diese Variable steuert, ob der Parser sich derzeit in einem aktiven oder inaktiven Teil eines IF-Blocks befindet. Diese Variable ist entweder 0 (IFTRUE) oder enthält die Tiefe der Verschachtelungsebene.

Die Verarbeitung der Bedingungen erfolgt durch PHP mit Hilfe der eval-Funktion:

```
@eval("\$this->intIfActive
    = \$strVar1Name \$strOperator \$strVar2Name
    ? 0 : \$this->intIfDepth;");
```

Als Operatoren sind deshalb alle Operatoren möglich, die PHP verarbeiten kann.

Teil eines IF-Blocks kann ELSE sein. Hier wird einfach der aktuelle Zustand umgedreht. Eine Prüfung auf den aktiven Teil erfolgt zuvor nicht, weil ELSE immer erkannt werden muss.

```
case 'ELSE':
    $this->check_closed($strTag);
    if ($this->intIfDepth > 0)
    {
        if ($this->intIfActive == IFTRUE)
        {
            $this->intIfActive = $this->intIfDepth;
        } elseif ($this->intIfActive == $this->intIfDepth)
        {
            $this->intIfActive = IFTRUE;
        }
    }
}
```

Der Fehler wird ausgelöst, wenn sich ELSE außerhalb eines IF befindet.

```
} else {
    trigger_error($this->objError->
        gen_msg(LSERR_TAGPOSITION,
            $this->ErrorData()),
        E_USER_WARNING);
}
break;
```

Variablenverarbeitung

phpTemple verwaltet eigene Variablen, darunter auch globale (Applikationsvariablen), wie sie von PHP nicht angeboten werden.

```
case 'SETGLOBAL':
case 'SET':
case 'UNSETGLOBAL':
case 'UNSET':
case 'VAR':
```

Prozessende

Am Ende des Parsers werden die Tags entfernt. Falls ein Tag so konstruiert ist, dass es sich selbst durch etwas anderes ersetzt, kann es dies in *\$strTagReplacement* ablegen. Verwendet wird es nur für *<LS:VAR>*, wo der Inhalt einer Variable zurückgegeben wird.

```
$strLineToParse = $strTagReplacement;
```

Der letzte Teil schreibt die Zeile, abhängig von IF, in den Ausgabepuffer. Überflüssige Leerzeichen und Umbrüche werden entfernt.

```
if ((int) $this->intIfActive == IFTRUE)
{
    if (strlen(chop($strLineToParse)) > 0)
    {
```

Variablen, die im HTML-Code stehen, werden nun auch ersetzt:

```
$strParsedLine = $this->Resolve_Parameter($strLineToParse);
```

Falls die Kodierung eingeschaltet wurde, werden die Codes mit Hilfe der Ersetzungstabelle *arrEncoding* ausgetauscht:

```
if ($this->blnEncoding)
{
    $strParsedLine=strtr($strParsedLine, $this->arrEncoding);
}
```

Die Kompression entfernt Leerzeichen, Tabulatoren und Zeilenumbrüche. Alleinstehende Leerzeichen müssen natürlich erhalten bleiben, mehrfache auf eines reduziert werden. Dies führt zu dem verhältnismäßig umfangreichen regulären Ausdruck.

```
if ($this->blnCompressing)
{
    $strParsedLine =
        preg_replace("/([\\n\\t]+)|({2,})|( )/U", '\\3',
            $strParsedLine);
}
```

Soll nicht komprimiert werden, sind die noch übrigen Zeilen wieder mit einem Zeilenumbruch zu versehen.

```
} else {
    $strParsedLine .= "\\n";
}
```

Die fertige Zeile wird nun dem Ausgabepuffer hinzugefügt. In einigen Fällen, wie bei der Bildung von Optionslisten, erzeugen Tags zusätzlichen Code (so entstehen aus einem `<select list="name">` gleich mehrere Zeilen mit `<option>`-Tags). Diese werden an die aktuellen Zeile angehängt (array_merge):

```
$this->arrOutputBuffer[] = $strParsedLine;
$this->arrOutputBuffer = array_merge($this->arrOutputBuffer,
    $this->arrGenerated);

unset($this->arrGenerated);
} /* end if */
} /* end if IF */
return TRUE;
} /* end method Parse_Line */
```

Einlesen einer Vorlage

Für das Einlesen einer Vorlage wird eine eigene Methode verwendet. Diese ist wenig spektakulär, die meisten Zeilen kümmern sich um das Abfangen von Fehlern.

```
function Read_Template($strTemplateName)
{
    $this->blnTemplateOK = FALSE;
```



```

if (file_exists($strTemplateName))
{
    $this->blnTemplateOK = TRUE;
    $this->strCurrentTemplate = basename($strTemplateName);
}

```

Der entscheidende Punkt ist das Einlesen der Vorlage in die Variable *arrCurrentSite*:

```

$this->arrCurrentSite = file($strTemplateName);
if (count($this->arrCurrentSite) == 0)
{
    $arrErr = $this->ErrorData();
    $arrErr['source'] = "$strTemplateName";
    trigger_error($this->objError->
        gen_msg(LSERR_TMPNOTFOUND, $arrErr),
        E_USER_ERROR);
    exit;
}
} else {
    $arrErr = $this->ErrorData();
    $arrErr['source'] = "$strTemplateName";
    trigger_error($this->objError->gen_msg(LSERR_TMPNOTFOUND,
        $arrErr), E_USER_ERROR);
    exit;
} /* end if */
return TRUE;
} /* end method Read_Template */

```

Verschlüsselung der URL

phpTemple verwendet eine einfache Verschlüsselung der URL. Damit werden Manipulationen stark erschwert und die Strukturierung der Vorlagen verschleiert. Das Verfahren basiert auf einer Base64-Codierung, kann aber bei Bedarf auf stärkere Verschlüsselungstechniken umgestellt werden. Diese dürften dann aber auch mehr Prozessorleistung benötigen.

```

function getcrypturl($strURL, $strQuotation)
{

```

Es kann ein Anführungszeichen (einfach oder doppelt) angegeben werden, das erhalten bleiben soll. Dieses darf nicht von PHP mit einem Backslash versehen werden, da HTML damit nichts anfangen kann:

```

$strQuotation = stripslashes($strQuotation);

```

Von der Verschlüsselung sind URLs ausgenommen, die auf externe Programme oder JavaScript verweisen:

```

if (!preg_match('#http://|mailto:|ftp://|javascript:|i',$strURL))
{

```

Dann wird die ursprüngliche URL zerlegt:

```
$arrURL = parse_url($strURL);
```

Zusätzlich wird die SessionID verpackt:

```
$arrURL['sid'] = session_id();
```

Falls Parameter folgen, werden diese serialisiert und kodiert:

```
if (strlen($arrURL['query']) > 0 )
{
    $strURL = $arrURL['host'] . $arrURL['path'] . '?' . LS_NS
        . '=' . base64_encode(serialize($arrURL));
}
```

Ansonsten bleibt die URL unverändert:

```
} else {
    $strURL = $arrURL['host'] . $arrURL['path'];
} /* end if */
} /* end if */
```

Am Ende werden die ursprünglichen Anführungszeichen wieder darum gesetzt:

```
return "$strQuotation$strURL$strQuotation";
} /* end function */
```

Die eigentliche Funktion zur Umwandlung der URL verwendet einen regulären Ausdruck, der in der inzwischen zur Zeichenkette gewandelten, fertig verarbeiteten Seite alle Elemente sucht, die mit href=, url= oder src= beginnen. Damit das funktioniert, müssen Anführungszeichen verwendet werden – egal ob einfache oder doppelte.

Die folgende Funktion gibt die fertige Seite auch sofort aus.

Ausgabe der Seite

```
function crypturl()
{
```

Zuerst wird der Inhalt des Ausgabepuffers gelesen. Die Ausgabe der Seite ist zuvor bereits in den Puffer erfolgt. Dieses Verfahren hat den Vorteil, dass die Verschlüsselung sich allein durch unterdrücken der Methode *crypturl* unterbinden lässt – ohne Änderungen am übrigen Code.

```
$strBuffer = ob_get_contents();
```

Dann wird der bisherige Puffer gelöscht.

```
ob_end_clean();
```

Es folgt der reguläre Ausdruck, der die Attribute href, url und src erkennt und deren Argumente unter Beibehaltung der Anführungszeichen mit der verschlüsselten Version ersetzt:

```
$strBuffer =
preg_replace('#(href|url|src)=(\x22){1}([^\x22]+)(\x22)?#ime',
```

```

        '"\1=".$this->getcrypturl("\3", "\2")',
        $strBuffer );
$strBuffer =
preg_replace('#(href|url|src)=(\x27){1}([^\x27]+)(\x27)?#ime',
        '"\1=".$this->getcrypturl("\3", "\2")',
        $strBuffer );

```

Damit die Session-ID auch in Formularen übertragen wird, wird ein verstecktes Feld dafür eingesetzt. Dies ist notwendig, weil `trans_sid` (die automatische Transformation in PHP) unterdrückt wird.

```

$strBuffer = preg_replace("#<form([>]*)>#im",
        '<form\\1><input type="hidden"
                                name="sid"
                                value="'
                                . session_id() . '" />',
        $strBuffer );

```

Als letztes wird der Puffer direkt an den Browser gesendet. Damit ist die eigentliche Verarbeitung der Seite beendet.

```

echo $strBuffer;
} /* end function crypturl() */

```

Weitaus weniger aufwändig ist der Start der Verarbeitung:

```

function Parse_Template()
{
    if ($this->blnTemplateOK)
    {
        $this->intLineCounter = 0;
    }
}

```

Die Schleife durchläuft einfach alle Zeilen der Vorlage und ruft die Methode `Parse_Line` für jede Zeile auf. Dort wird eventuell das Array `arrCurrentSite` und der Zeiger `intLineCounter` manipuliert – vor allem um eingeschlossene Dateien einbinden und Schleifen bilden zu können. Deshalb ist es nicht möglich, hier `foreach` zu verwenden.

```

while ($strLineToParse =
        $this->arrCurrentSite[$this->intLineCounter++])
{
    $this->Parse_Line($strLineToParse);
} /* end while */

```

Anschließend wird die Pufferung eingeschaltet:

```
ob_start();
```

Wurden von Tags HTTP-Header erzeugt, werden diese der Seite vorangestellt:

```

if ((strlen($this->strHeader) > 0) and (!headers_sent()))
    header ($this->strHeader);

```

Dann werden HTML-Header ausgegeben:

```
echo $this->strPageStart;
```

Dann wird das fertige Seitenarray in den Ausgabepuffer gesendet:

```
echo implode(' ', $this->arrOutputBuffer);
```

Wurde festgelegt, dass die Seite im Cache gespeichert werden soll, ist diese Stelle eine gute Idee. Auch hier wird einfach der Ausgabepuffer gelesen:

```
if ($this->blnCacheFile)
{
    $hdlF = fopen($this->strCacheFile, 'w');
    fwrite($hdlF, ob_get_contents());
    fclose($hdlF);
}
```

Für den Teil, der sofort ausgegeben werden soll, erfolgt nun die Verschlüsselung der URLs:

```
$this->crypturl();
```

Die Seite steht nun wieder – verändert – im Ausgabepuffer. Mit dem nächsten Befehl wird sie an den Browser gesendet.

```
ob_end_flush();
return TRUE;
} else {
return FALSE;
}
```

11.6.4 Formularsteuerung

phpTemple verwendet eine ausgereifte Formularsteuerung, die viele Aktionen, die zur Gestaltung komfortabler Formulare notwendig sind, übernimmt. Ein Blick auf die Klasse *LS_formvalidator* zeigt, wie dies funktioniert.

Erkennung und Analysieren von Formularfeldern

Damit Felder automatisch verarbeitet werden können, muss *phpTemple* alle Felder, die per POST oder UPLOAD vorliegen, analysieren. Die Prüfinformationen sind dagegen an die Session gebunden und werden im Array *arrSVars* bereit gestellt. Weitere wichtige Variablen sind:

- *\$this->arrPostVars*

Das Array mit allen gesendeten Feldern

- *\$this->arrPostFiles*

Das Array mit allen gesendeten Dateien

```
function field_read ()
{
    if (is_array($this->arrPostVars))
    {
        $bInFieldError = FALSE;
    }
}
```

Der folgende Teil durchläuft alle Felder:

```
foreach($this->arrPostVars as $strVarName => $varVarValue)
{
```

Zuerst wird sichergestellt, dass die interne Verwaltung von PHP für die Felder unterdrückt wird.

```
unset($GLOBALS[$strVarName]);
```

Wenn ein Prüfwert für ein Feld existiert, wird die Prüfmethode *field_proof* aufgerufen:

```
if (isset($this->arrSVars["PROOF_$strVarName"]))
{
    if ($this->field_proof($strVarName, $varVarValue,
                        $this->arrSVars["PROOF_$strVarName"]))
    {
```

Liegt kein Fehler vor, wird die Fehlerinformation gelöscht:

```
unset($this->arrSVars['ERROR_' . $strVarName]);
    } else {
```

Andernfalls wird das globale Fehler-Flag gesetzt:

```
        $bInFieldError = TRUE;
    } /* end if */
} /* end if */
```

Jetzt werden die verschiedenen Feldarten verarbeitet. Wenn ein Kontrollkästchen angeklickt wurde, wird eine interne Variable *{CHECKED_name}* mit dem Wort *checked* belegt. Der Entwickler kann also alleine durch setzen dieser Variable bestimmen, dass der Wert beim erneuten Aufruf des Formulars erhalten bleibt.

Kontrollkästchen

```
if (strlen($this->arrSVars["CHECKED_$strVarName"]) > 0 )
{
    $this->arrSVars["CHECKED_$strVarName"] = 'checked';
}
```

Ganz ähnlich funktioniert das mit Optionsschaltflächen.

Optionsfelder

```
if (isset($this->arrSVars["RADIO_$strVarName"]))
{
    $this->arrSVars['RADIO_' . $strVarName . '_' . $varVarValue]
        = 'checked';
}
```

Etwas aufwändiger ist die Verarbeitung der Listenelemente. Eine mögliche Mehrfachauswahl wird als kommaseparierte Liste übergeben. Aus praktischen Gründen wurde nämlich vom Parser die sonst übliche Ergänzung mit [] für Mehrfachfelder entfernt.

```
if (is_array($varVarValue))
{
    $this->arrSVars["SELECT_$strVarName"]
        = implode(',', $varVarValue);
    foreach ($varVarValue as $strSelectValue)
    {
        $this->arrSVars['SELECTED_' . $strVarName . '_'
            . $strSelectValue] = 'selected';
    }
    $this->arrSVars["FIELD_$strVarName"] = $varVarValue;
}
```

Listenelemente

Für alle anderen Felder wird der Wert in einer Variablen mit dem Präfix FIELD abgelegt.

```
} else {
    $this->arrSVars["FIELD_$strVarName"]
        = stripslashes($varVarValue);
    $this->arrSVars['SELECTED_' . $strVarName . '_'
        . $varVarValue] = 'selected';
}
} /* end foreach */
```

Dateiupload automatisieren

Eine der interessantesten Funktionen ist der automatische Dateiupload. Hier genügt es, in einem Formularfeld entsprechende Attribute zu ergänzen: *phpTemple* legt die Dateien automatisch ab und erzeugt gegebenenfalls entsprechende Fehlermeldungen. Es ist für den Entwickler keine einzige Zeile Code notwendig.

```
if (is_array($this->arrPostFiles))
{
```

Mit Hilfe der Schleife ist die Anzahl der Dateien unbegrenzt:

```
foreach($this->arrPostFiles as $strFieldName => $arrPostFiles)
{
    if ($arrPostFiles['size'] > 0)
    {
```

Zuerst werden die Werte ermittelt, die PHP beim Hochladen erkannt hat.

```
$strFileName      = $arrPostFiles['name'];
$intFileSize      = (int) $arrPostFiles['size'];
$strFileMIME      = $arrPostFiles['type'];
$strFileTemp      = $arrPostFiles['tmp_name'];
```

Dann werden die Zielwerte aus den vom Parser ermittelten gelesen:

```

$strFileTargetDir
    = $this->arrSVars['UPLOADDIR_' . $strFieldName];
$strFileTargetMIME
    = $this->arrSVars['UPLOADMIME_' . $strFieldName];
$intFileTargetSize
    = (int) $this->arrSVars['UPLOADSIZE_' . $strFieldName];
$blnSuccess = TRUE;

```

Anschließend werden die Zielwerte mit den tatsächlichen verglichen und entsprechende Fehlermeldungen erzeugt: Hier ist noch eine Ergänzung zu erwarten: Die Fehlermeldungen sollten nicht hardcodiert sein.

```

if ($intFileSize > $intFileTargetSize)
{
    $this->arrSVars['UPLOADSIZEERROR_' . $strFieldName] =
        "Upload fehlgeschlagen: Die hochgeladene Datei war zu
        groß (<u>$intFileTargetSize</u> Byte erlaubt,
        <u>$intFileSize</u> Byte geladen).";
    $blnSuccess = FALSE;
}
if (strstr($strFileTargetMIME, $strFileMIME) === FALSE)
{
    $this->arrSVars['UPLOADMIMEERROR_' . $strFieldName] =
        "Upload fehlgeschlagen: Dateityp nicht akzeptiert
        (<u>$strFileTargetMIME</u> erwartet,
        <u>$strFileMIME</u> gesendet)";
    $blnSuccess = FALSE;
}

```

Traten keine Fehler auf, kann die bislang im temporären Verzeichnis befindliche Datei an den Zielort kopiert werden:

```

if ($blnSuccess)
{
    $this->arrSVars['UPLOADFULL_' . $strFieldName]
        = dirname($PATH_TRANSLATED)
        . "$strFileTargetDir/$strFileName";
    $this->arrSVars['UPLOADNAME_' . $strFieldName]
        = $strFileName;
    $this->arrSVars['UPLOADSIZE_' . $strFieldName]
        = $intFileSize;
    $this->arrSVars['UPLOADMIME_' . $strFieldName]
        = $strFileMIME;
    @copy ($strFileTemp, dirname($PATH_TRANSLATED)
        . "$strFileTargetDir/$strFileName");
} /* end if */
} else {
    $this->arrSVars['UPLOADERERROR_' . $strFieldName]
        = "Upload fehlgeschlagen: Es wurden keine Daten
        übertragen.";
} /* end if */

```

```

    } /* end foreach */
} /* end if */

```

Steuerung der Folgetemplates

phpTemple besitzt eine interne Verwaltung der Vorlagen. Deshalb ist die Steuerung von Zielen besonders einfach möglich. Der folgende Teil der Methode *field_read* ermittelt das Ziel und setzt die Variable `${TEMPLATE}` entsprechend.

Im ersten Teil wird versucht, Felder mit den Namen `FOLLOWSITE_ERROR` bzw. `FOLLOWSITE_OK` zu erkennen. Diese werden vom Parser auf Grundlage des Tags `<LS:FOLLOW>` als versteckte Felder erzeugt.

```

if ($bInFieldError and
    (isset($this->arrPostVars['FOLLOWSITE_ERROR'])))
{
    $this->arrSVars['TEMPLATE']
        = $this->arrPostVars['FOLLOWSITE_ERROR'];
} else if (isset($this->arrPostVars['FOLLOWSITE_OK']))
{
    $this->arrSVars['TEMPLATE']
        = $this->arrPostVars['FOLLOWSITE_OK'];
}

```

Für Mehrfachverzweigungen über Schaltflächen werden die Namen der Absendeschalter ausgewertet. Auch diese Elemente erzeugt der Parser aus `<LS:FOLLOW>`. Für die angeklickte Schaltfläche existiert ein Wert in `${FIELD_schaltflaechenname}`:

```

if (is_array($this->arrPostVars['FOLLOWSITE_SUBMIT']))
{
    foreach($this->arrPostVars['FOLLOWSITE_SUBMIT']
        as $strTemplate => $strSubmitName)
    {
        if (strlen($this->arrSVars["FIELD_$strSubmitName"]) > 0)
            $this->arrSVars['TEMPLATE'] = $strTemplate;
    }
}

```

11.6.5 Datenbankzugriff

Der Datenbankzugriff ist nicht direkter Bestandteil von *phpTemple*. Eingesetzt wird die hervorragende Bibliothek ADODB. Sie finden die aktuellste Version unter der folgenden Adresse:

<http://php.weblogs.com/ADODB>

Mit ADODB unterstützt *phpTemple* sehr viele Datenbanken, beispielsweise MySQL, PostgreSQL, Oracle, MS SQL Server, Sybase und alle über ODBC erreichbaren Datenbanken (beispielsweise Access). Allerdings wurde *phpTemple* bislang nur mit MySQL praktisch getestet.

Nach Einbinden der entsprechenden Bibliotheken stellt ADODB folgende Objekte zur Verfügung:

- `ADOConnection`

Ein Objekt, dass die Verbindung zur Datenbank steuert.

- `ADORecordSet`

Ein Objekt, dass Datensätze verwaltet.

11.6.6 Installation und Administration

Um mit *phpTemple* arbeiten zu können, muss eine Datenbank existieren. Diese Daten und die Verzeichnisinformationen werden in die Datei `LS.INI` eingetragen. Dann starten Sie die Datei `INSTALL.HTM` und folgen den Anweisungen. Das Skript erstellt dann eine neue `CONFIG.INC.PHP4` und legt die nötigen Tabellen an. Wenn die Tabellen schon existieren, werden diese nicht neu aufgebaut, damit keine Daten verloren gehen. Bereits angelegte Kundentabellen bleiben generell unberührt.

Unmittelbar nach der erfolgreichen Erstellung der Konfigurationsdatei kann mit der – derzeit noch manuellen – Einrichtung begonnen werden. Dies umfasst folgende Schritte:

- Erstellen Sie die Vorlagen und kopieren Sie diese in die Verzeichnisse `/TEMPLATE` bzw. `/INCLUDE`.
- Tragen Sie mit Hilfe eines Datenbankadministrationsprogramms wie *phpMyAdmin* die Templates in die Tabelle `xx_linkresolve` ein. Für `xx` tragen Sie den gewählten Systempräfix ein.
- Starten Sie die Applikation durch Aufruf der Datei `INDEX.PHP4` ohne Parameter. Die Standardvorlage `INDEX.HTM` wird aufgerufen (soweit Sie diese Vorgabe nicht verändert haben).

11.7 phpTemple-Projekte im Web

In diesem Abschnitt werden zwei Projekte vorgestellt, die mit *phpTemple* bisher realisiert wurden. Beiden ist gemeinsam, dass die Umsetzung der HTML-Vorlagen innerhalb kürzester Zeit, das heißt weniger Tage, vollzogen werden konnte. Die Steigerung der Produktivität gegenüber klassisch programmierten PHP-Projekten betrug mehr als 500%.

11.7.1 Mauerfotos

Dieses Projekt ist eine Bildverwaltung mit 488 Schwarz/Weiß-Fotos der Berliner Mauer, entstanden zwischen 1984 und 1987. Die Aufgabe bei der Realisierung der Website bestand in der Umsetzung einer Benutzerführung, die sowohl eine schnelle Navigation als auch gute Interaktion bot.

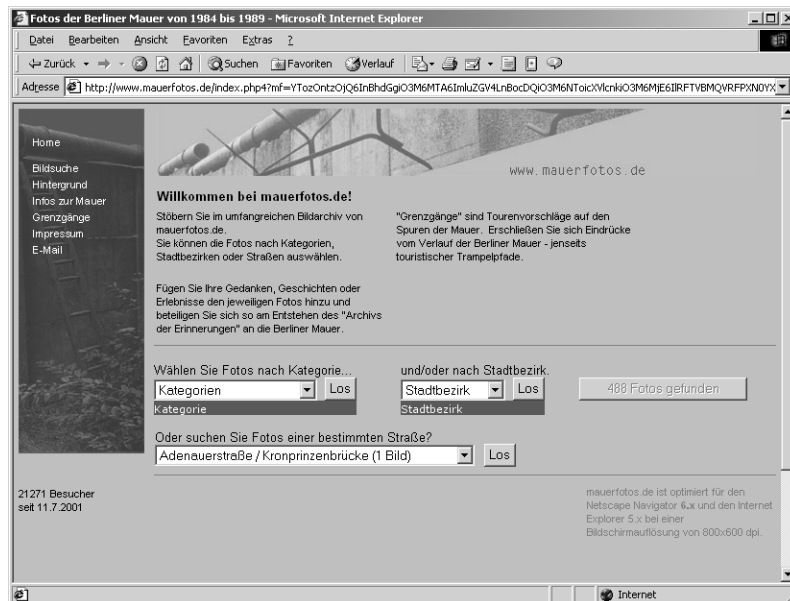


Abbildung 11.5:
Auswahl der
Bildkategorien

Die Website finden Sie unter folgender Adresse:

<http://www.mauerfotos.de>

Besonders intensiv werden die Listen- und Navigationsfunktionen genutzt.

Skriptbeispiele aus www.mauerfotos.de

Die Besonderheit der Benutzerführung besteht in der gegenseitigen Abhängigkeit der beiden Auswahllisten *Kategorien* und *Stadtbezirke*. Wenn der Nutzer eine Auswahl auf einer Seite trifft, wird die andere Liste soweit reduziert, wie auch tatsächlich Auswahlmöglichkeiten bestehen. Dazu wird die Abfrage der Listen mit einer Bedingung gesteuert, die folgendermaßen erzeugt wird.

Wenn der Benutzer eine Kategorie auswählte, ist die Feldvariable `{FIELD_kategorie}` gesetzt.

```
<LS:IF VAR1="{FIELD_kategorie}" OP="!=" VAR2="">
  <LS:SET VAR="countcondition"
```

```
VALUE="cat_id = ${FIELD_kategorie}"/>
<LS:SET VAR="bezirkcondition" VALUE="0"/>
```

In einer Schleife wird jetzt die Bildtabelle nach allen Fotos durchsucht, die in der ausgewählten Kategorie stehen. Durch DISTINCT wird dies auf die Liste der Bezirke reduziert.

```
<LS:DATA LOOPS="0"
    STMT="SELECT DISTINCT CONCAT(',',',', bezirk_id)
        AS bezirkid
        FROM mf_fotos
        WHERE cat_id = ${FIELD_kategorie}">
```

Aus den Daten wird fortlaufend eine kommaseparierte Liste der vorhandenen Bezirke erstellt:

```
<LS:SET VAR="bezirkcondition"
    VALUE="${bezirkcondition}${bezirkid}"/>
</LS:DATA>
```

Für die weitere Auswahl wird daraus eine IN-Anweisung erstellt, die später die Auswahl reduziert.

```
<LS:SET VAR="fotocondition"
    VALUE="bezirk_id IN (${bezirkcondition})"/>
<LS:SET VAR="bezirkcondition"
    VALUE="id IN (${bezirkcondition})"/>
```

Wurde keine Einschränkung angegeben, wird als Auswahlkriterium 1 erzeugt, damit die Listenauswahl funktioniert:

```
<LS:ELSE/>
    <LS:SET VAR="bezirkcondition" VALUE="1"/>
</LS:IF>
```

Damit korrespondiert dann die Auswahl der Listen, hier am Beispiel der Liste der Stadtbezirke:

```
<select size="1" runat="server" name="bezirk">
    <option value="">Stadtbezirk</option>
    <LS:DATA LOOPS="0" STMT="SELECT listoption AS distoption,
        listvalue AS distvalue
        FROM mf_lists
        WHERE listlanguage='de'
        AND listtype='bezirk'
        AND ${bezirkcondition}">
        <option ${AUTOSELECTED} runat="server"
            value="${distoption}">${distvalue}</option>
    </LS:DATA>
</select>
```

Ausgabe der Vorschauseite

Auf der Vorschauseite werden verkleinerte Bilder angezeigt. Die Forderung bestand in der Ausgabe von maximal 15 Bildern pro Seite in fünf Spalten. Der folgende Code realisiert dies.

```
<LS:DATA LOOPS="0" NAVIGATION="15"
  STMT="SELECT nr AS fotoid, datum AS date,
    IF(LENGTH(titel) > 50,
      CONCAT(LEFT(titel, 47), '...'), titel) AS title,
    nailname AS thumb
  FROM mf_fotos
  WHERE ${distcondition}
  AND ${catcondition}
  AND ${streetcondition} ">
```

Die SQL-Anweisung erledigt alleine die eigentliche Arbeit. Beachten Sie die Verkürzung des Textes der Bildtitel mit IF. Dabei werden alle Texte mit mehr als 50 Zeichen (LENGTH) auf 47 Zeichen gekürzt und mit drei Punkten ergänzt (CONCAT).

Um die Seite zur Anzeige der Bilder in voller Größe zu erreichen, ist ein entsprechender Link notwendig. Die Informationen über die Auswahl werden hier mitgeschleift, damit die Auswahl nicht verloren geht:

```
<a href="${LINK_picturedeu}&id=${fotoid}
  &INFO_OFFSET=${INFO_OFFSET}&FIELD_bezirk=${FIELD_bezirk}
  &FIELD_kategorie=${FIELD_kategorie}
  &FIELD_streetselect=${FIELD_streetselect}
  &FIELD_street=${FIELD_street}">
  
</a>
```

Für die nächsten Spalten wird dann einfach nur ein Datensatz weitergeschaltet:

```
<LS:DATANEXT/>
```

Unterhalb der Auswahl soll eine Navigation erscheinen. Zuerst die Anzeige der Anzahl der Fotos:

```
<LS:IF NUM1="${INFO_SUM}" OP=">" NUM2="1">
  &nbsp;Fotos <b>${INFO_FROM}</b> bis <b>${INFO_TO}</b>
  von <b>${INFO_SUM}</b>
</LS:IF>
```

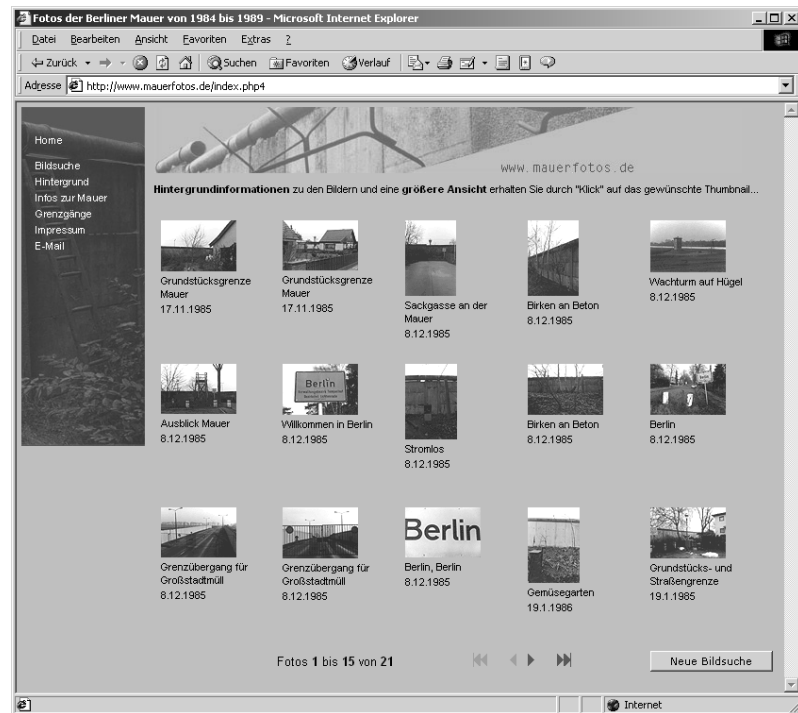
Dann folgt die Erzeugung der Schaltflächen zur Navigation zwischen den Seiten:

```
<LS:IF NUM1="${NAVI_FIRST}" OP="==" NUM2="${INFO_OFFSET}">
  
<LS:ELSE/>
  
</LS:IF>

```

Die anderen Navigationselemente sind ähnlich aufgebaut, anstatt der Variablen `${NAVI_FIRST}` wird jedoch `${NAVI_NEXT}` usw. verwendet. Genutzt wird hier ein `onclick`-Ereignis, ebenso gut könnte man mit `<a href>` arbeiten.

Abbildung 11.6:
Ausgabe der
Vorauswahl auf
mehreren Seiten



Das gesamte Projekt konnte übrigens ohne eine einzige Zeile PHP-Code realisiert werden. Die Vorlagen können nun weiterverarbeitet oder angepasst werden, ohne dass Eingriffe in die Programmierung erforderlich sind. An vorbereitenden Arbeiten war der Entwurf der Datenbank und der SQL-Abfragen erforderlich, was sicher umfangreicheres Verständnis erfordert.

11.7.2 Weitere Projekte

Mit *phpTemple* sind bereits eine ganze Reihe von Projekten realisiert worden. Die bisherigen Projekte umfassen:

- Eine Suchmaschine

- Ein profilgesteuertes Auswahlprogramm für Webseiten
- Diverse Shopsysteme
- Verwaltung der Website zu diesem Buch

Wenn Sie schnell ein Projekt auf der Basis von *phpTemple* umsetzen möchten, steht der Autor gern zur Verfügung. Wenn fertige HTML-Dateien vorliegen, ist eine datenbankgestützte Funktionalität in wenigen Tagen realisierbar.

11.8 phpTemple für professionellen Einsatz

Wenn Sie beabsichtigen, selbst mit *phpTemple* zu entwickeln und Kunden dann das Komplettsystem zu verkaufen, sollten Sie sich die professionelle Variante ansehen. Im Gegensatz zum vorgestellten Open Source-Projekt unterliegt diese Version – *phpTempleX* – einer anderen Lizenz und basiert auf etwas anderen Grundsätzen.

11.8.1 Lizenzrechtliche Fragen

Open Source-Projekte, die meist unter der GPL veröffentlicht werden, verlangen meist, dass auch die Weitergabe diese Lizenzform beibehält. Basiert das eigene Geschäft jedoch primär auf dem Verkauf von Software und nicht auf der damit verbundenen Dienstleistung, hilft die GPL wenig. Kommerzielle Software kostet zwar Geld, kann aber ebenso gut für Geld verkauft werden. Entsprechend gibt es mit *phpTempleX* eine Version, die Ihnen das Recht gibt, das Programm selbst weiterzuverkaufen – zu jedem beliebigen Preis.

Das Lizenzmodell sieht eine einfache Form der Verbreitung vor: Sie erwerben zu einem einmaligen Pauschalpreis den Quellcode zur nicht exklusiven und zeitlich unbeschränkten Nutzung. Sie können daraus so viele Kopien wie erforderlichlich anfertigen und diese beliebig oft und zu jedem beliebigen Preis verkaufen. Die Höhe des Lizenzpreises richtet sich nach dem beabsichtigten Einsatz und dem Umfang der künftigen Nutzung.

11.8.2 Programmtechnische Modifikationen

phpTempleX enthält einige Features, die den Einsatz für größere Projekte prädestinieren:

- Vorlagen müssen in XHTML 1.0 (oder höher) geschrieben sein und werden entsprechend der XHTML-DTD verarbeitet und geprüft.

- Die Einschränkung, dass Tags zeilenweise angeordnet sein müssen, besteht nicht mehr.
- Prozessanweisungen sind einfacher integrierbar.
- Die Verschlüsselungsfunktion des URL ist abschaltbar, außerdem kann die Art der Verschlüsselung verändert werden.
- SSL wird auf allen Plattformen unterstützt.
- Die Verarbeitung ist deutlich schneller.
- Eine umfangreiche Administration zum Aufsetzen eines Projekts mit wenigen Mausklicks auf beliebigen Servern wird mitgeliefert.
- Netzwerklastverteilungssysteme werden unterstützt.

11.8.3 Weitere Eigenschaften

Wie bei kommerziellen Produkten üblich, gehören einige Leistungen zum Lieferumfang:

- Der Code ist mit dem Zend-Encoder geschützt, eine Lizenz des Encoders wird mitgeliefert, sodass Sie ihre Modifikationen wieder schützen können.
- Zum Lieferumfang gehört die aktuelle PHP-Version mit den benötigten Modulen und Zend-Launchpad zur einfacheren Installation auf allen verfügbaren Betriebssystemen (entspricht der Verfügbarkeit von PHP auf den verschiedenen Plattformen selbst).
- Lieferung auf CD und mit gedrucktem Handbuch.
- Kostenloser E-Mail-Support für 180 Tage ab Kaufdatum.
- Kostenloses Update, wenn innerhalb eines Jahres nach Kaufdatum eine neue Hauptversion erscheint. Weiterhin verbilligte Updates für jedes beliebige weitere Update – auch wenn die Updates nicht kontinuierlich bezogen werden.
- Serviceverträge sind verfügbar.

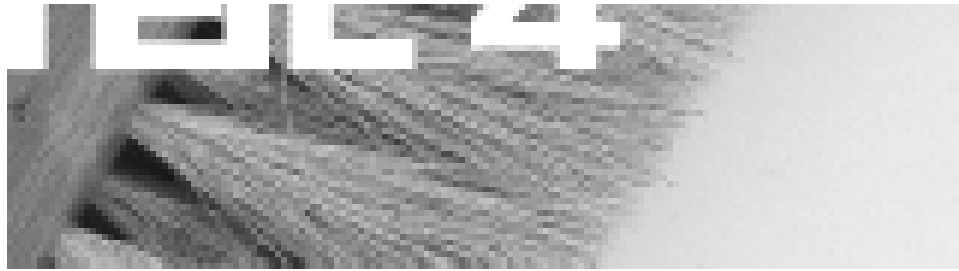
11.8.4 Kontaktadresse

Die erhalten weitere Informationen, Angebote und Preisangaben auf der Webadresse zum Projekt *phpTemple*:

<http://www.phptemple.de>

Des weiteren können Sie natürlich jederzeit den Autor direkt unter seiner E-Mail-Adresse *joerg@krause.net* ansprechen.

TEIL IV



Anhänge und
Referenz

A Glossar

A

Access Control

Zugangskontrolle. Bestimmt, wer Zugang zu den Ressourcen eines Computers oder einer Applikation hat.

ACL

Access Control List (Windows). Die Zugangskontrollliste ist eine Liste mit der kontrolliert wird, welche Nutzer zu welchen Diensten Zugang haben. **Access Control List**

Active Group, Die

Eine Standardisierungsorganisation unterhalb der Open Group. Schwerpunkt ist die Standardisierung und Lizenzierung der ActiveX-Technologie. Informationen finden Sie unter <http://www.activex.org>.

Active Server

Eine Sammlung von serverseitigen Technologien, die mit dem Windows NT-Server geliefert werden. Dazu gehören Komponenten, Skriptmodelle und ein integrierter Satz von Systemdiensten für die Datenbanksteuerung, Transaktionssteuerung und Nachrichtendienste.

Active Server Pages (ASP)

Eine serverseitige Skriptumgebung, die ActiveX-Skripte und ActiveX-Komponenten benutzt. Skripten und Komponenten können zu Applikationen für das Web kombiniert werden.

ActiveX

Ein Oberbegriff von Microsoft für alle Technologien, die es Entwicklern ermöglichen, interaktive Anwendungen für das Web zu programmieren. Dazu gehört ein Satz von sprachunabhängigen Technologien, die es erlauben, in verschiedenen Sprachen geschriebene Komponenten in einer Netzwerkumgebung zusammen auszuführen. Die Kerntechnologien sind das Component Object Model (COM) und das Distributed Component Object Model (DCOM). Diese Technologien werden durch die Open Group lizenziert. *Siehe auch* → COM, → CGI, → DCOM, → Java.

ActiveX-Automation

Ein sprachunabhängiger Weg zur Manipulation von ActiveX-Komponenten außerhalb der Applikation. Der Begriff wurde früher OLE-Automation genannt.

ActiveX-Komponente

Ein kompiliertes Stück Software, basierend auf der → COM-Technologie. ActiveX-Komponenten können sowohl auf dem Server als auch dem Client ausgeführt werden. ActiveX-Komponenten können in Scriptsprachen wie VBScript oder JScript geschrieben werden.

V

S

I

1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E

Java-Applets werden automatisch zu ActiveX-Komponenten mit der Erweiterung .CLASS, wenn sie von der Java Virtual Machine ausgeführt werden. ActiveX-Komponenten sind auch als DLLs .DLL, ActiveX-Controls .OCX oder ausführbare Programme .EXE möglich. Siehe auch → COM und → DCOM.

ActiveX-Object

Ein Objekt, das anderen Applikationen oder Programmierwerkzeugen über das ActiveX-Automation Interface bereitgestellt wird.

ActiveX-Serverkomponente

Eine ActiveX-Komponente, die für einen Server entworfen wurde und Bestandteil einer Client-Server-Applikation ist. Siehe auch → ActiveX-Control.

Adabas D

Adabas ist eine relationale Datenbank, die auf vielen Plattformen zur Verfügung steht.

ADO

Active Data Objects. Eine Sammlung von Objekten, die den Zugriff auf Datenbanken erlauben und für die Verwendung in internetbasierten Applikationen optimiert sind. ADO wird mit dem Internet Information Server und Visual Studio geliefert.

Aggregation

Eine Technik, die bei der Implementierung von Objekten die Verwendung anderer Objekte und deren Eigenschaften und Methoden unterstützt.

Alias

Ein Name, der einen Teil der URL auf einem physischen Pfad auf dem Webserver abbildet. Wird vor allem zur Zusammenfassung langer physischer Pfade und zur Organisation von Pfaden benutzt.

ALT

Parameter im HTML-Tag , der einen Text zur Anzeige bringt, wenn das zugeordnete Bild nicht angezeigt wird oder der Mauszeiger über dem Bild schwebt.

Anchor (A)

Anker. Quelle oder Ziel eines Hypertext-Links.

Anonymous FTP

Teil des → FTP (File Transfer Protocol). Der Zugriff erfolgt anonym, ist also für jeden Nutzer offen. Die Anmeldung erfolgt durch den Nutzernamen »anonymous«; als Kennwort wird meist eine E-Mail-Adresse akzeptiert, teilweise ist keine Eingabe möglich.

ANSI

American National Standards Institute. Eine Standardisierungsorganisation.

**American National
Standards Institute**

ANSI-Zeichensatz

Zeichensatz, der vor allem in Windows-Umgebungen benutzt und von der Standardisierungsorganisation ANSI unterstützt wird.

Apache

Kostenlos verfügbarer Webserver, der auf viele Plattformen portiert wurde und inzwischen am häufigsten eingesetzt wird. Der Name stammt von »a patchy server«.

Apache-Modul

Funktionserweiterungen für den → Apache-Webserver, die direkt eingebunden werden. Ein wichtiges Modul ist → PHP.

API

Application Programming Interface. Ein Satz von Unterprogrammen auf niedrigem Niveau, mit denen Dienste des Betriebssystems bereitgestellt werden. Es wird auch ein Satz von Vorschriften darunter verstanden, nach dem Dienste von Applikationen benutzt werden müssen.

**Application
Programming
Interface**

Applet

Ein in HTML-Seiten eingebettetes Programmstück, das in der Sprache → Java geschrieben wurde und temporär vom Server auf den Client übertragen wird.

Application

Ein Computerprogramm, eine Anwendung oder eine Gruppe von ASP-Skripten, die eine bestimmte Aufgabe erfüllen.

Application root

Das Stammverzeichnis einer → Applikation, unterhalb dessen sich alle Dateien und Ordner befinden.

Argument

Eine Konstante, Variable oder ein Ausdruck, das an eine Prozedur übergeben wird.

Array

Ein Satz sequenzieller Datenelemente, bestehend aus demselben Datentyp. Jedes Element eines Arrays hat eine eindeutige Identifikationsnummer – den Index. Änderungen an einem Element beeinflussen die anderen Elemente nicht. Wird auch als Feld bezeichnet.

ARP

Address Resolution Protocol. Verbindet dynamisch die physische Netzwerkadresse eines lokalen TCP/IP-Netzwerks mit der IP-

**Address
Resolution
Protocol**

V
S
I
1
2
3
4
5
6
7
8
9
10
11

A

B

C

D

E

Nummer der Hosts. Sie ist in der RFC 826 definiert. Siehe auch → TCP/IP.

ASCII

American Standard Code for Information Interchange

American Standard Code for Information Interchange. Ein 7-bit-Zeichensatz, der auf fast allen Computersystemen verbreitet ist. ASCII enthält 128 Zeichen. Diese 128 Zeichen entsprechen dem umfangreicheren → ANSI-Zeichensatz.

ASCII-Datei

Eine Bezeichnung für eine reine Textdatei, die nur ASCII-Zeichen enthält. Neben Buchstaben, Zahlen und Satzzeichen gehören auch Zeilenumbrüche, Wagenrücklaufcodes und Tabulatoren dazu, aber keine programmspezifischen Sonderzeichen. Textverarbeitungsprogramme wie Word erzeugen keine reinen ASCII-Dateien, wenn es nicht ausdrücklich verlangt wird. Formatierungen sind in ASCII-Dateien nicht möglich.

ASP

Active Server Pages

Skriptumgebung für Microsoft-Webserver. ASP ist offen für mehrere Programmiersprachen, beispielsweise → VBScript, → JScript, → Perl oder Rexx.

ATM

Asynchronous Transfer Mode

Asynchronous Transfer Mode. Ein Kommunikationsprotokoll für Hochgeschwindigkeitsdatenübertragungen.

Attribute

Informationen, die eine Datei oder einen Befehl näher beschreiben. Im Windows NT-Dateisystem können Dateien die Attribute »Nur Lesen«, »Versteckt«, »System«, »Komprimiert« oder »Archiv« zugewiesen werden.

Auditing

Protokollieren von ausgewählten Aktivitäten bestimmter Nutzer in einer Logdatei auf dem Server.

Authentication – Authentifizierung

Kontrolle der Identität eines Nutzers, basierend auf bestimmten Nutzerinformationen. Typischerweise wird dazu ein Name und ein Kennwort verwendet.

Authorization – Autorisierung

Die Autorisierung beschreibt das Recht einer einzelnen Person oder einer Gruppe von Personen, sich an einem lokalen oder entfernten Computer bestimmter Ressourcen – Dateien und Daten – zu bedienen. Die Autorisierung wird von einem Systemadministrator vergeben. Sie wird durch eine → Authentifizierung erkannt.

Automation

Eine Technologie, die es einem Programm auf dem Client-Computer erlaubt, ein Objekt zu erzeugen und zu kontrollieren.

B**Bandbreite**

Die Kapazität eines Übertragungsmediums in Bits pro Sekunde oder als Frequenz in Hertz. Eine höhere Bandbreite bedeutet, dass mehr Daten pro Sekunde übertragen werden können.

Basic authentication

Ein einfaches Autorisierungsprotokoll, das zur Absicherung von Webseiten dient und Kennwörter im Klartext überträgt.

Baud

Die Geschwindigkeit, mit der Modems Daten übertragen. Normalerweise ist 1 Baud gleich 1 Bit pro Sekunde (bps). Die Bezeichnung Bit pro Sekunde ist exakter, da Baud die Möglichkeit der Datenübertragung, Bit pro Sekunde jedoch die wirklich übertragenen Daten beschreibt.

BIND

Siehe → DNS.

Boolean

Ein Variablentyp, typischerweise 1 oder 0. Boolesche Variablen werden oft in logischen Ausdrücken verwendet, um einen Wahrheitswert zu ermitteln, der wahr oder falsch sein kann. Boolesche Ausdrücke entstehen, wenn Boolesche Variablen mit logischen Operatoren (Und, Oder, Nicht usw.) verknüpft werden.

Boolescher Ausdruck

Ein logischer Ausdruck, der TRUE (Wahr) oder FALSE (Falsch) sein kann.

bps

Bits per second. Bits pro Sekunde. Siehe → Baud.

Bits per second

Browser

Eine Software auf einem Client-Computer, die im HTML-Format erstellte Dateien darstellen kann. Browser können auch clientseitige Skriptsprachen wie Javascript oder VBScript ausführen.

Bytecode

Die ausführbare Form von Javacode, der von der Java Virtual Machine ausgeführt wird. Wird auch Pseudo-Code genannt.

C**C, C++**

Höhere Programmiersprache. C++ ist eine Weiterentwicklung, die sich vor allem durch → objektorientierte Funktionen auszeichnet.

V
S
I
1
2
3
4
5
6
7
8
9
10
11

A

B
C
D
E

Cache

Schneller Speicher für Dateien, die häufig benutzt werden. Dateien im Cache sind möglicherweise nicht aktuell, wenn sie angefordert werden.

Certificate Authority (CA)

Eine Organisation, die Zertifikate herausgibt, verwaltet und einzieht. Solche Organisationen verstehen sich als vertrauenswürdige Dritte, werden also von beiden Partnern einer Kommunikationskette anerkannt. Eine bekannte Firma ist VeriSign.

CGI**Common Gateway Interface**

Common Gateway Interface. Ein serverseitiges Interface, das Software-Dienste zur Verfügung stellt, oder auch ein Satz Interfaces, die beschreiben, wie ein Webserver mit anderer Software auf einem Computer kommuniziert. Jede Software kann ein CGI-Programm sein, wenn es Daten nach CGI-Standard herausgibt oder empfängt. Auch → PHP kann als CGI-Programm betrieben werden.

CGIBIN-Verzeichnis

Das Verzeichnis eines Webserver, in dem CGI-Programme abgelegt sind, die ausgeführt werden. Dieses Verzeichnis muss wenigstens über Skript-Rechte am Webserver verfügen.

CGI-Script

Ein Programm, das mit dem Webserver kommuniziert. Wenn beispielsweise ein Nutzer ein Formular absendet, werden die Daten von einem CGI-Script verarbeitet.

Client

Eine Applikation oder ein Prozess, der Dienste von anderen Prozessen oder Komponenten auf Servern anfordert. In einer Client-Server-Umgebung kann das auch ein kompletter Computer sein, beispielsweise ein PC mit Browser. In einer COM-Umgebung ist der Client ein Objekt, das Dienste von anderen Objekten anfordert.

Client-Server-Architecture

Ein Computermode, wo → Client-Applikationen auf einem Computer Informationen und Dienste von entfernten Computern oder Servern anfordern. Der Client ist in diesem Modell für die Interaktion mit dem Nutzer, der Server ist auf die Bereitstellung von Daten für viele gleichzeitig zugreifende Nutzer optimiert.

Collision detection

Siehe → CSMA/CD.

COM**Component Object Model**

Component Object Model. Ein Modell für objektorientierte Programmierung, das definiert, wie Objekte innerhalb einer Applikation oder zwischen Applikationen interagieren. In COM greift die Client-Software auf Objekte über Zeiger auf Interfaces zu. Das

Software auf Objekte über Zeiger auf Interfaces zu. Das Interface wird durch einen Satz an Funktionen gebildet, die hier Methoden genannt werden.

Component

Englisch für → Komponente.

Content type

Typ einer Datei (beispielsweise Text, Grafik, Sound), der durch die Dateierweiterung identifiziert wird (.txt, .gif, .wav usw.).

Control

In einer grafischen Benutzerumgebung ein Objekt, das auf dem Bildschirm dargestellt wird und an dem der Nutzer Eingaben vornehmen kann. Typische Controls sind Schalter, Rollbalken, Kontrollkästchen usw.

Cookie

Cookies dienen der Speicherung von Informationen durch einen Server oder eines Skripts auf einem Server auf dem Client-Computer. Cookies sind kleine Textdateien, die Bestandteil des Browsers sind. Cookies enthalten beispielsweise die Sessionvariablen in ASP-Skripten. Die Informationen innerhalb der Cookies können nur von dem Server oder Skript abgerufen werden, der sie auch erzeugt hat.

CORBA

Common Object Request Broker Architecture. Eine Spezifikation der Object Management Group, die ein Interface zwischen OMG-kompatiblen Objekten definiert.

Crawler

Siehe → Spider, → Robot.

CSMA/CD

Carrier Sense Multiple Access with Collision Detection. In paketorientierten Netzwerken, wie beispielsweise Ethernet, ein Mechanismus, mit dem die sendende Station sicherstellt, dass sich auf einem Strang nicht zwei Pakete gleichzeitig bewegen. Jeder Computer, der Daten senden möchte, horcht zuerst auf der Leitung und sendet erst dann, wenn die Leitung frei ist. Entscheiden zwei Computer gleichzeitig über den Sendebeginn, wird die entstehende Kollision ausgewertet, beide Stationen warten nun eine durch einen Zufallsgenerator definierte und damit unterschiedlich lange Zeitspanne und beginnen erst dann erneut auf eine freie Leitung zu warten und zu senden.

Cursor

Ein kleines Stück Software, das Spalten einer Tabelle an eine Applikation übergibt. Ein Cursor auf einer Ergebnisliste identifiziert die bestimmte Position innerhalb der Liste.

**Carrier Sense
Multiple Access
with Collision
Detection**

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

In einem grafischen Benutzerinterface kennzeichnet der Cursor die Position des Zeigege­r­ätes, normalerweise der Maus. In einer textorientierten Umgebung bildet der Cursor die Schreibmarke, an der die nächste Eingabe von der Tastatur erscheint.

D

Daemon

Ein Programm unter Unix, das im Hintergrund arbeitet und auf bestimmte Ereignisse reagiert. Der Name stammt von »Disc and Execution Monitor«.

Datagramm

Eine selbstständige Einheit von Daten, die von einem Punkt in einem Netzwerk zu einem anderen Punkt übertragen wird. Siehe auch → Frame, → Paket.

Data Source, Datenquelle

Der Name, den Applikationen benutzen, um eine Verbindung über das Open Database Connectivity (ODBC)-Interface aufzubauen. Die Datenquelle spezifiziert den Namen des Computers und den → Data Source Name (DSN).

Data Source Name

Data Source Name Name einer Datenquelle. Möglich sind File DSN, User DSN und System DSN. File DSNs beschreiben die Schnittstelle in einer Datei und stehen dem lokalen Nutzer auf seinem Computer zur Verfügung. User DSN stehen dem zugeordneten NT-Benutzer zur Verfügung, während System-DSN allen Teilnehmern des Netzwerkes zur Verfügung stehen.

DCOM

Distributed Component Object Model Distributed Component Object Model. Erweiterung des Component Object Model (COM), die die transparente Verteilung von Objekten über das Netzwerk oder über das Internet unterstützt. DCOM ist durch die Open Group spezifiziert.

Deadlock

Eine Situation, in der sich zwei oder mehrere Skripten gegenseitig permanent blockieren. Wenn beispielsweise Skript A den Datensatz 1 sperrt, bis Datensatz 2 freigegeben wird, und gleichzeitig Skript B den Datensatz 2 sperrt, bis Datensatz 1 freigegeben wird, kommt es zu einem Deadlock – beide Skripte warten ewig. Ein Deadlock ist jedoch kein Systemabsturz, der Taskmanager wird keinen Fehler erkennen können.

Debugger

Ein Softwarewerkzeug, das Fehler im Quellcode eines Programms oder Skripts finden kann oder den Entwickler bei der Fehlersuche durch geeignete Funktionen unterstützt. Wesentliches Merkmal ist

eine schrittweise Programmausführung und Anzeige der Variableninhalte.

Default Document

Kann im IIS definiert werden. Der IIS wird das Default Document dann an den Browser senden, wenn in der URL kein Dateiname spezifiziert wurde.

Default Gateway

Das Standard-Gateway in einem TCP/IP-Netzwerk. Als Default Gateway wird eine Netzwerkadresse interpretiert, an die Clients ihre Pakete senden, wenn die Zieladresse außerhalb des eigenen Netzwerks liegt. Das Gateway wird die Daten dann an das übergeordnete Netzwerk weiterleiten.

DES

Data Encryption Standard. Ein Verschlüsselungsprotokoll, das das Ausspähen von Kennwörtern verhindern soll. Der Microsoft Remote Access Service (RAS) benutzt DES zur Sicherung des Verbindungsaufbaus.

Data Encryption Standard

Design Time

Die Zeit, in der ein Entwickler die Applikation zusammenstellt, programmiert, die Oberfläche und die Komponenten entwirft. Im Gegensatz dazu gibt es die → Run Time (deutsch Laufzeit).

DFÜ-Netzwerk

Baustein in Windows, mit dem die Anwahl (→ Dial-up) eines entfernten Computers, eines Internet Service Providers oder eines entfernten Windows-PC gesteuert wird. Die Nutzung des DFÜ-Netzwerks setzt voraus, dass eine entsprechende Hardware zur Herstellung der entfernten Verbindung vorhanden ist (Netzwerkkarte, ISDN-Karte, Modem).

DHCP

Dynamic Host Configuration Protocol. Ein Standardprotokoll, das der Zuweisung von IP-Konfigurationen dient. Dazu wird im Netzwerk ein DHCP-Server eingerichtet. Windows NT enthält die nötigen Komponenten für einen DHCP-Server.

Dynamic Host Configuration Protocol

DHTML

Dynamic HTML. Ein Funktionssatz, der die Sprache HTML um dynamische Elemente erweitert. Dabei kann die mit DHTML erstellte Webseite mit dem Nutzer ohne Zugriff auf den Server interagieren. Die sprachlichen Elemente basieren auf JavaScript; zur Adressierung der Elemente einer Seite wird ein Objektmodell verwendet.

Dynamic HTML

Dial-up

Der Vorgang des Verbindungsaufbaus durch einen Computer über eine Telefonleitung. Benutzt wird dabei ein Modem.

V
S
I
1
2
3
4
5
6
7
8
9
10
11

A

B

C

D

E

Digitale Signatur

Teil eines digitalen Zertifikats, der einen Verschlüsselungscode enthält, der den Inhaber des Zertifikats eindeutig identifiziert.

Directory Browsing

Eine Funktion, die das Durchsuchen eines Verzeichnisses auf dem Webserver mit Hilfe eines Browsers erlaubt. Sie sollten diese Funktion abschalten, denn das Sichtbarmachen der Verzeichnisstrukturen stellt ein Sicherheitsloch dar.

DLL**Dynamic Link Library**

Dynamic Link Library. Eine Bibliothek mit ausführbaren Funktionen, auf die andere Programme zugreifen können. DLLs sind ein Bestandteil des Betriebssystems Windows. Die Funktionssammlungen werden als Datei mit der Erweiterung .DLL gespeichert. Code kann so mehrfach verwendet werden.

DNS**Domain Name System**

Domain Name System. Ein Protokoll und die Art und Weise, wie im Internet die IP-Adressen mit leichter verständlichen Namen verbunden werden. Die Umsetzung übernehmen so genannte Nameserver oder DNS-Server, die für jede IP-Adresse einen entsprechenden Eintrag vorhalten. DNS-Server arbeiten hierarchisch. Kann ein DNS-Server die Anfrage nicht beantworten, leitet er sie an den übergeordneten Server bis zu Root-Servern weiter, die von den Network Information Centers (NIC) betrieben werden. Die NIC vergeben Domainnamen.

DNS Name**Domain Name System**

Domain Name System. Siehe → Domainname, → DNS.

DNS Nameserver

Im DNS-Client-Server-Modell ein Computer, der Informationen über einen Teil der DNS-Datenbank enthält und Anfragen von Computern aus dem Internet über die Zuordnung von IP-Nummern zu Domainnamen beantwortet. Siehe → DNS.

DNS reverse lookup

Prozess zum Ermitteln des Namens anhand einer gegebenen IP-Nummer.

DNS spoofing

Fälschung eines DNS-Namens eines anderen Systems durch Manipulation eines Nameservers oder auch die unzulässige Simulation eines Nameservers für zulässige Domains.

Domainname

Teil der Domain Name System (DNS)-Namensstruktur. Der Begriff bezeichnet einen Namen im Internet, bestehend aus Servername, Domainname und Topleveldomain. Die Topleveldomain bezeichnet ein

Land (beispielsweise DE, AT etc.) oder ein generisches Level, beispielsweise COM.

DSN

→ Data Source Name.

Dynamic HTML

→ DHTML.

Dynamische Webseite

Eine Webseite, die erst nach bestimmten Aktionen des Nutzers erzeugt wird.

E**E-Commerce**

Electronic Commerce. Der Prozess des Kaufens und Verkaufens von Waren und Leistungen über das Internet oder andere Online-Medien. Im weitesten Sinne der elektronische Handel.

Electronic Commerce

E-Mail

Bezeichnung für den Austausch von Nachrichten und Informationen über elektronische Medien, lokale Netzwerke, das Internet oder proprietäre Netzwerke und Dienstleister. E-Mail ist der populärste Dienst des Internets.

Ethernet

Ein Standard für lokale Netzwerke mit einer Übertragungsrate von Brutto 10 Mbit/sec. Benutzt wird die → CSMA/CD-Zugriffssteuerung.

Ereignis, Event

Eine Aktion, die durch den Nutzer oder ein ActiveX-Control erzeugt wurde und das Programm zu einer entsprechenden Interaktion veranlasst. Als Ereignis wird beispielsweise ein Mausklick verstanden; die Interaktion kann das Aufrufen einer weiteren Webseite sein, wenn der Klick auf einen Link erfolgte.

Exception, Ausnahme

Eine unnormale Bedingung oder ein Fehler, der während der Programmausführung auftrat und die Software zu einer geänderten Programmausführung veranlasst. Programme, die für Ausnahmebedingungen nicht vorbereitet wurden, stürzen ab.

Expires header

Das Verfallsdatum einer Datei, die im Cache oder Proxy gehalten wird und mit dem der Zeitpunkt beschrieben wird, an dem die Daten der Datei ungültig werden und erneut durch eine Kopie des Originals ersetzt werden müssen.

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Extranet

Ein → Intranet, an dem auch externe Personen oder Firmen beteiligt sind.

F**Frequently Asked Questions****FAQ**

Frequently Asked Questions. Häufig gestellte Fragen. Ein Dokument mit einer Liste von besonders häufig gestellten Fragen mit den dazugehörigen Antworten. Durch FAQ-Listen sollen Anfänger oder Neulinge in die Lage versetzt werden, sich sehr schnell mit einer Thematik vertraut zu machen, ohne »immer wieder dieselben Fragen zu stellen«.

FilePro

Ein relationales Datenbanksystem.

Filter

Im Internet Information Server eine ISAPI-Funktion, die den Datenstrom zum Webserver *vor* der Verarbeitung durch den Webserver und *nach* der Verarbeitung, jedoch vor dem Absenden, bearbeiten kann.

Finger

Ein Programm, das Informationen zu bestimmten Nutzern anzeigt, die sich an einem lokalen oder entfernten System angemeldet haben. Neben dem Namen werden auch die An- und Abmeldezeiten und Angaben über die Arbeitsstation angezeigt, wenn diese Daten verfügbar sind.

Firewall

Ein System oder eine Reihe von Systemen, mit denen ein internes, lokales Netzwerk vor Zugriffen von außerhalb geschützt werden soll. Firewalls filtern Pakete im TCP-Datenstrom und entscheiden anhand der Adresse über die Weiterleitung.

Footer, Fußzeile

Bezeichnung für das Anfügen eines kurzen HTML-Codes am Ende jeder Webseite, die gesendet wird. Wird vom Webserver gesteuert und kann vom Entwickler der Seite nicht beeinflusst werden.

Form, Formular

Bezeichnung für den Teil einer HTML-Seite, in dem der Nutzer Daten eingeben kann, die an den Webserver zurückgesendet werden.

Fully qualified domain name**FQDN**

Fully qualified domain name. Siehe → Domainname.

Frame

In einem TCP/IP-Netzwerk das komplette Paket, das übertragen wird, bestehend aus Adressinformationen, Parametern sowie den eigentlichen Daten; zusammengefasst entsteht ein Frame.

In HTML ein Teil des Browserbildschirms, dessen Inhalt durch eine eigene Datei gesteuert wird. Frames sind Bestandteil von HTML 4 und dienen der besseren Bildschirmkontrolle und -aufteilung.

FreeBSD

Ein Unix-Dialekt, der an der Berkeley Universität entwickelt wurde. Er ist ebenso frei wie → Linux, aber weniger verbreitet.

FrontPage-Servererweiterungen

Einige Dateien, die auf einem Webserver installiert werden, um Nutzern bestimmte Funktionen zur komfortablen Verwaltung von Web-Projekten zur Verfügung zu stellen. Zur Benutzung müssen Sie über Frontpage oder Visual InterDev (VID) verfügen.

FTP

File Transfer Protocol. Ein Protokoll des Internets, das der schnellen Übertragung von Dateien dient.

**File Transfer
Protocol**

G**Gateway**

Eine Hardware oder Software, die an eine bestimmte Adresse gerichtete Datenpakete erkennt und weiterleitet. Gateways können auch das Medium wechseln, beispielsweise ein lokales → Ethernet-Netzwerk mit einer ISDN-Leitung verbinden.

GDBM

Der → GNU-Datenbankmanager.

GET

Methode zur Datenübertragung unter → HTTP.

GIF

Graphics Interchange Format. Ein im Internet weitverbreitetes Grafikformat, das Mitte der 80er Jahre von Compuserve entwickelt wurde. GIF komprimiert verlustfrei mit nur 256 Farben und eignet sich vor allem für Schrift und Schaltflächen.

**Graphics
Interchange Format**

GNU

GNU steht für »Gnu ist nicht Unix«. Eine Form des Softwarevertriebs als Open Source, bei der die Quellen von Grundlagensoftware frei verteilt werden.

Gopher

Ein frühes Internetprotokoll. Zeigt Dokumente auf entfernten Computern an; Vorläufer des WWW.

GUI

Graphical User Interface. Der Teil der Software, der für die Darstellung der Informationen in grafischer Form auf dem Bildschirm verantwortlich ist.

**Graphical User
Interface**

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

H**Handshake**

Ein Begriff für die Art und Weise, wie Computer miteinander Informationen austauschen.

Header

Informationen, die am Anfang eines HTTP-Prozesses übertragen werden.

Hit-Zähler

Ein Programm, Skript oder eine interne Funktion, mit denen die Zugriffe auf eine Webseite gezählt werden.

Homepage

Die Stammseite eines Webs, der Startpunkt einer Internetpräsentation. Oft auch ein allgemeiner Begriff für die Existenz einer solchen Darstellung.

Host

Ein Computer, der in einem Netzwerk anderen Computern Dienste anbietet. Allgemein jeder Computer, der mit irgendeiner Art Netzwerk verbunden ist.

Hostname

Der Name eines Computers, der als → Host arbeitet.

HTML**Hypertext Markup Language**

Hypertext Markup Language. Eine einfache Beschreibungssprache für Dokumente, die plattformunabhängig arbeitet. HTML-Dateien sind einfache ASCII-Dateien mit eingefügten Codes, die durch Marken (Tags) indiziert werden. HTML-Dateien enthalten auch → Hyperlinks. HTML ist die Formatiersprache für Web-Dokumente. Siehe auch → DHTML, → SGML, → XML.

HTTP**Hypertext Transfer Protocol**

Hypertext Transfer Protocol. Das zugrunde liegende Protokoll, mit dem Webserver und Webclients (→ Browser) kommunizieren. HTTP arbeitet auf Applikationsniveau und ist für die Verteilung und Funktion von multimedialen Dokumenten zuständig. HTTP ist verbindungslos und objektorientiert. HTTP kann Verbindungen aufbauen, ohne dass unbedingt Daten übertragen werden müssen.

Hyperlink

Ursprung des Wortes Link. Damit wird eine Verbindung zu einer anderen Stelle im Internet oder einer anderen Stelle eines → Hypertext-Dokuments bezeichnet. Hyperlinks werden oft durch Unterstreichungen oder durch die Veränderung des Mauszeigers zu einer Hand vom übrigen Text abgesetzt. Durch Anklicken des Hyperlinks wird die verknüpfte Adresse ausgewählt.

Hypertext, Hypertext-Dokument

Ein Dokument, das Links (Hyperlinks) enthält und in dem Inhalte miteinander durch Hyperlinks verknüpft sind. Praktisch ist fast jede HTML-Datei ein Hypertext-Dokument.

I**ICMP**

Internet Control Message Protocol. Eine Erweiterung des Internet-Protokolls (IP). ICMP erlaubt die Erzeugung von Fehlermeldungen, Testpaketen und informellen Nachrichten. Siehe auch → PING.

**Internet Control
Message Protocol**

IETF

Internet Engineering Task Force. Eine Organisation, welche die Entwicklung und Standardisierung von Protokollen und Diensten für das Internet vorantreibt. Die IETF ist eine internationale und offene Organisation von Netzwerkentwicklern und -designern, Herstellern, Betreibern und Forschungseinrichtungen, deren Aufgabe die Entwicklung der Internetarchitektur und des Betriebs des Internets ist. Informationen finden Sie unter <http://www.isoc.org/>.

**Internet
Engineering Task
Force**

Image-Map

Eine Grafik, die zusätzliche Informationen zu bestimmten → URLs enthält und die beim Anklicken einer bestimmten Position den Browser zum Aufruf des Links veranlasst.

IMAP

Protokoll zur Übertragung von E-Mail. Erlaubt die Speicherung von persönlicher E-Mail auf dem Server und damit die entfernte Verwaltung.

**Internet Message
Access Protocol**

Index-Datei

Das Standarddokument eines Webs. Wird üblicherweise »index.htm« oder »index.html« genannt.

Inheritance

Siehe → Vererbung.

In-Prozess-Komponente, in-process component

Eine so bezeichnete Komponente läuft im Prozessraum des Client. Üblicherweise eine DLL. Im Gegensatz dazu gibt es Out-Prozess-Komponenten.

Instanz, instance

Ein Objekt, das von der Klasse einer Komponente abgeleitet wurde. Jede Instanz hat eigene Adressräume für Variablen und arbeitet unabhängig von anderen Instanzen, die von der gleichen Klasse abgeleitet wurden. Der Begriff Objekt wird als Synonym für »Instanz einer Klasse« benutzt.

V
S
I
1
2
3
4
5
6
7
8
9
10
11

A

B

C

D

E

Instanzieren

Der Prozess, eine → Instanz eines Objekts zu erzeugen.

Interaktive Application

Ein Programm oder Skript, geschrieben in PHP, VBScript, C, PERL usw., das durch einen Hyperlink ausgelöst wird.

Interface, Schnittstelle

Eine Gruppe logischer Operationen und Methoden, die Zugriff auf ein Objekt oder auf Komponenten erlauben.

Internet

Zusammenschluss vieler Computer und Computernetzwerke zum Datenaustausch unter dem Protokoll TCP/IP und darauf aufbauender Protokolle.

Internet-Dienst

Jedes typische Protokoll außer HTTP. Dienste sind unter anderem Gopher, Telnet, WAIS, NNTP, FTP.

InterNIC

Internet Network Information Center. Das derzeitige Anmeldezentrum für die Topleveldomains .COM, .NET, .ORG, .EDU, .GOV und .MIL. Adresse: <http://internic.net>.

Interoperabilität

Die Fähigkeit von Hard- und Software verschiedener Computersysteme verschiedener Hersteller, miteinander zu kommunizieren.

Intranet

Die Nutzung der Technologien des Internet für ein verteiltes privates Netzwerk.

IP**Internet Protocol**

Internet Protocol. Teil des TCP/IP, das Nachrichten von einem Ort zu einem anderen transportiert. IP sendet TCP-Pakete und enthält keine höheren Funktionen zur Kontrolle der Integrität der Daten oder zur Verschlüsselung.

IP-Adresse

Internet Protocol-Adresse. Eine einmalige und eindeutige Nummer, die einen Computer im Netzwerk identifiziert. Die Adresse ist 32 Bit lang und wird mit 4 Dezimalzahlen geschrieben, beispielsweise: 62.208.3.33.

ISAPI

Internet Server Application Program Interface. Eine Schnittstelle, die auf einem Internet Information Server zur Verfügung gestellt wird, um Anwendungsprogrammen bestimmte Funktionen zur Verfügung zu stellen.

ISDN

Integrated Services Digital Network. Digitaler Telefoniedienst, der auch Datenübertragungsdienste bis 128 KBit zur Verfügung stellt.

**Integrated Services
Digital Network**

ISO

International Organization for Standardization. Eine Organisation, die 1946 gegründet wurde, um internationale Industriestandards zu vereinbaren, die auch Computer und Kommunikation betreffen.

ISP

Internet Service Provider. Ein Unternehmen, das einen Einwahlknoten für die temporäre Verbindung zum Internet bereitstellt. Der ISP selbst verfügt über permanente Verbindungen auf der Basis von Standleitungen, die ihn mit anderen ISPs verbinden.

**Internet Service
Provider**

J**Java**

Ein Derivat der Sprache C++, das als offener Standard für Internetanwendungen von der Firma SunSoft entwickelt wurde.

Java Virtual Machine (Java VM)

Ein Subsystem zu einem Betriebssystem oder Browser, unter dem Java-Programme ablaufen können.

JavaBeans

Ein Objektmodell der Firma SunSoft, das interoperable Funktionen unter Java bereitstellt – einschließlich COM und CORBA.

JavaScript

Eine Skriptsprache mit ähnlicher syntaktischer Struktur wie Java, entwickelt von Netscape. Verwendet HTML-Seiten als Nutzerschnittstelle.

JDBC

Java Database Connectivity. Schnittstelle für den Datenbankzugriff unter Java.

**Java Database
Connectivity**

JPEG, JPG

Joint Photographic Experts Group. Grafikformat, das von allen Browsern verstanden wird. Die Komprimierung nach diesem Standard ist variabel und kann verlustbehaftet sein. Bis zu 16,7 Millionen Farben werden verarbeitet. Das Format wird vor allem für die Komprimierung von Bildern benutzt. JPG ist die Dateierweiterung für Dateisysteme nach der 8.3-Konvention.

**Joint Photographic
Experts Group**

JScript

Microsofts Implementierung der Sprache → JavaScript.

V
S
I
1
2
3
4
5
6
7
8
9
10
11

A

B
C
D
E

K**KBps**

Kilobytes pro Sekunde.

Key

Schlüssel

Wird in der Windows-Registrierung und für die Absicherung von verschlüsselten Webseiten verwendet. In PHP der Index eines assoziativen Arrays.

Key pair, Schlüsselpaar

Die Kombination eines privaten und eines öffentlichen Schlüssels zur Sicherung der Daten für die Übertragung über ein unsicheres Netzwerk.

Klasse

Die formale Definition eines Objekts. Die Klasse ist eine Art Vorlage, von der weitere → Instanzen des Objekts abgebildet werden.

Klassen-ID (class ID = CLSID)

Eine universelle eindeutige Nummer, die eine COM-Komponente identifiziert.

Kommunikationsprotokoll

Ein Befehlssatz, nach dessen Regeln der Austausch von Daten zwischen Computern stattfindet. Beispiele sind HTTP, TCP/IP und SNA.

Komponente

Eine eindeutige Einheit aus ausführbarem Code, der mit der ActiveX-Technologie entwickelt wurde. Von Komponenten werden durch Instanziierung Objekte abgeleitet.

Kryptographie

Wissenschaft von der Verschlüsselung.

L**LAN**

Local Area Network

Local Area Network. Lokales Netzwerk aus Computern.

LDAP

**Lightweight
Directory Access
Protocol**

Lightweight Directory Access Protocol. Ein Protokoll, das Verzeichnisdienste bereitstellt.

Link

Siehe → Hyperlink.

Linux

Frei verfügbares Betriebssystem auf der Basis von Unix. Wird international von vielen freien Entwicklern weiterentwickelt. Linux hat im Bereich der Webserver eine große Verbreitung gefunden.

localhost

Ein Ersatzname für den lokalen Computer.

Logdatei, Protokolldatei

Eine Datei, in der Vorgänge auf einem Computer gespeichert werden. Protokolle können in Textdateien oder Datenbanken gespeichert sein. Sie enthalten Reaktionen von Programmen oder Aktionen von Benutzern. Die Protokolle des Webservers können benutzt werden, um die Aktivität des eigenen Webs zu kontrollieren.

logging

Protokollieren. Der Vorgang, Daten zu erfassen und in eine Logdatei zu schreiben.

Logisches Laufwerk

Ein eigenständig ansprechbarer und benannter Teil einer Festplatte, der sich physisch den Platz mit weiteren logischen Einheiten teilt. Logische Laufwerke werden in erweiterten → Partitionen der Festplatte gespeichert.

M**MAPI**

Mail or Messaging Applications Programming Interface. Ein offener Standard für den Nachrichtenaustausch, beispielsweise E-Mail, Terminplaner, Kalender, Dokumentenmanagement.

**Mail or Messaging
Applications
Programming
Interface**

Marshaling

Der Vorgang oder Prozess, mit dem Parameter gepackt und über die Grenzen eines Prozesses oder Threads hinaus gesendet werden.

MD5

Eine Verschlüsselungsmethode, die im Internet benutzt wird.

message passing

Eine Methode, mit der parallel laufende Prozesse miteinander kommunizieren können.

message queuing

Eine Servertechnologie, mit der große Applikationen entwickelt werden können. Applikationen wird damit die Möglichkeit gegeben, trotz unterbrochener Verbindungen weiter zu funktionieren. Der Nachrichtenaustausch findet über einen Message Queue Server statt.

Metabase

Eine Struktur, in welcher der Internet Information Server Informationen über die Konfiguration ablegt. Die Funktionsweise ist ähnlich der Registry, benötigt aber weniger Speicher.

Metadaten

Metadaten sind Daten, die andere Daten beschreiben. Der Index Server benutzt beispielsweise Metadaten, um andere Daten so zu be-

V
S
I
1
2
3
4
5
6
7
8
9
10
11

A

B
C
D
E

schreiben, dass eine effiziente Suche stattfinden kann, ohne auf die Originaldaten zugreifen zu müssen.

Methode

Eine Prozedur oder Funktion, die in einem Objekt gespeichert ist und diesem Objekt zur Verfügung steht.

**Management
Information Base****MIB**

Management Information Base. Eine Software, welche die Aspekte eines Netzwerkes beschreibt, das mit → SNMP (Simple Network Management Protocol) gemanagt wird.

middle tier, Mittlere Schicht

Die mittlere Schicht einer Mehr-Schicht-Umgebung (multi-tier), auch Applikationsschicht genannt; die Schicht zwischen der Nutzerschnittstelle (beispielsweise dem Web-Client) und der Datenbank. Typischerweise liegen auf dieser Ebene der Webserver und damit auch PHP-Anwendungen.

Middleware

Eine Systemsoftware in Netzwerkumgebungen, die zwischen der Applikationsebene, dem Betriebssystem und der Netzwerktransportebene liegt. Als Middleware werden beispielsweise die Verzeichnisdienste, Transaktions-Prozesse, Object Request Broker, Remote Procedure Call-Dienste und Datenbankschnittstellen bezeichnet.

**Multipurpose
Internet Mail
Extension****MIME, MIME mapping**

Multipurpose Internet Mail Extension. Eine Möglichkeit, Browsern Informationen über Dokumente mitzuteilen, die mehrere Formate beinhalten.

multi-tier architecture, Mehr-Schicht-Architektur

Wird oft auch als Drei-Schicht-Architektur bezeichnet. Generell ist damit eine Technik gemeint, bei der Anwendungssoftware in drei Teile gesplittet wird: Nutzerschnittstelle, Applikation, Datenzugriff. Solche Applikationen basieren auf einem Objektmodell, beispielsweise ActiveX.

Multithreading

Schneller sequenzieller Ablauf verschiedener Prozesse eines Programms, unabhängig von der Multitasking-Methode des Betriebssystems. Der Nutzer bekommt den Eindruck, Programmteile würden parallel ablaufen; subjektiv wird eine Geschwindigkeitssteigerung empfunden.

mSQL

Schlankes und einfaches Datenbanksystem für kleine Projekte.

MSSQL

Komplexes und professionelles Datenbanksystem von Microsoft. Ermöglicht die Umsetzung großer Anwendungen. Verfügt über hervorragende Entwicklungswerkzeuge.

MySQL

Freies und kostenloses Datenbanksystem, das einen großen Funktionsumfang auf einfachem Niveau realisiert. Durch die einfache Implementierung, die auf viele Sicherheitsfeatures wie bei → MSSQL verzichtet, ist MySQL sehr schnell. Eignet sich für kleine und mittlere Projekte.

N**Namensauflösung**

Eine Methode, IP-Adressen leicht merkbare Namen zu geben. Siehe → DNS.

NCSA

National Center for Supercomputing Applications. Zentrum der Grundlagenforschung für Webanwendungen an der Universität Urbana, USA. Vom NCSA stammt der erste Webserver, der NCSA-Daemon.

network sniffer, Netzwerk-Schnüffler

Ein Werkzeug aus Hard- und Software, mit dem Kennwörter erkannt und analysiert werden können. Wird oft zur Fehlersuche benutzt. Werden Kennwörter im Klartext übertragen, können sie mit solchen Werkzeugen gelesen werden.

NNTP

Network News Transfer Protocol. Das Protokoll, mit dem Nachrichtengruppen verteilt und zwischen News-Server (beispielsweise IIS 4) und News-Clients (beispielsweise Outlook Express) übertragen werden. Die Spezifikation finden Sie in der RFC 977.

**Network News
Transfer Protocol**

Node, Knoten

Als Knoten wird jeder Computer bezeichnet, der an ein Netzwerk angeschlossen ist. Ein anderer Name dafür ist → Host.

O**Objekt**

In der → objektorientierten Programmierung die Bezeichnung für eine Variable, die sowohl Daten als auch ausführbaren Code enthält und als eine Einheit verwaltet wird. Objekte basieren auf bestimmten Modellen, die dem Client die Nutzung der Methoden (der im Objekt befindlichen Funktionen) erlauben und die Technik des Zugriffs auf die enthaltenen Daten beschreiben.

V

S

I

1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E

Objektorientierte Programmierung (OOP)

Der neueste Versuch, die reale Welt in einer Computerumgebung abzubilden. Unter der OOP wird ein übergreifendes Konzept verstanden, das Technologien zur Entwicklung von Softwareprodukten so beschreibt, dass die Komponenten extrem modular und wiederverwendbar sind. Anwendungsprogramme, Daten, Netzwerke und Computerressourcen erscheinen als Objekte, die gemischt eingesetzt werden können und im Ergebnis nicht mehr fester Bestandteil einer Applikation sind.

Open Database Connectivity**ODBC**

Open Database Connectivity. Eine Schnittstelle, die es Applikationen erlaubt, Daten von einer Vielzahl unterschiedlicher Datenquellen zu beschaffen. ODBC ist vom Betriebssystem unabhängig.

Object Linking and Embedding**OLE**

Object Linking and Embedding. Eine Sammlung verschiedener Standards, welche die Übertragung und Verteilung von Informationen zwischen Anwendungsprogrammen beschreibt. Ein Protokoll, das die Erzeugung kombinierter Dokumente beschreibt; diese Dokumente enthalten Informationen zu den Anwendungsprogrammen, die für die Bearbeitung zuständig sind. OLE basiert auf → COM.

OLE DB

Eine Datenbankschnittstelle für SQL- und Nicht-SQL-Datenbanken.

Object Management Group**OMG**

Object Management Group. Eine Gruppe Softwarehersteller, die die CORBA-Objektspezifikation definieren und vertreten.

Open Group, The

Eine Mutterorganisation für eine Anzahl Standardisierungsorganisationen, einschließlich The ActiveX Group, X/Open und OSF.

Oracle

Professionelles und umfangreiches Datenbanksystem der Firma Oracle Corp.. Oracle eignet sich für die größten Projekte und ist auf vielen Plattformen verfügbar.

Object Request Broker**ORB**

Object Request Broker. Software, die Interaktionen zwischen Client und Server steuert, einschließlich unter den für verteilte Computersysteme geltenden Bedingungen. Der ORB ist auch für die Koordinierung von Parametern und Ergebnissen zuständig.

Out-of-Process component

Eine ActiveX-Komponente, die in einem von der Applikation getrennten Speicherbereich läuft. Der Microsoft Transaction Server (MTS) kann Komponenten, die als DLL vorliegen, so verwalten, dass sie als Out-of-Process-Komponente benutzt werden können.

P**Paket, packet**

Eine Übertragungseinheit mit einem festen maximalen Umfang binärer Informationen, die sowohl Daten als auch Kopfinformationen und die Absende- und Zieladresse, Fehlerkorrekturinformationen und ID-Nummern enthalten kann. Kurz: ein Stück Daten, das über das Netzwerk gesendet wird.

Page, Seite

Siehe → Webseite.

Parser

Eine Software, die Text interpretiert. Der Text kann ein Quelltext (wie C++) sein oder eine Dokumentenbeschreibungssprache (wie HTML). Der Parser kontrolliert den Text auf syntaktische und semantische Fehlerfreiheit und übergibt den »geparsten« Text, meist in einem effizienten und kompakten internen Code, an die verarbeitende Einheit.

Partition

Teil einer Festplatte, der logisch gebildet wird und sich wie eine weitere Festplatte verhält.

PDF

Spezielles, postscriptähnliches Format für die plattformunabhängige Darstellung von Dokumenten. Entwickelt von der Firma Adobe. PDF-Dateien können mit Adobe Acrobat oder → PHP erzeugt werden.

Portable Document Format

Pfad, physischer

Ein vollständiger Pfad zu einer Datei, welcher der Universal Naming Convention (UNC) entspricht.

Pfad, relativer

Ein Teil eines physischen UNC-Pfades mit Platzhaltern. Der relative Pfad kann nur exakt bestimmt werden, wenn die Ausgangsposition des Pfades bekannt ist.

Pfad, URL

Damit ist eine vollständige → URL gemeint, die auch den Weg zu einer bestimmten Datei oder einem Verzeichnis durch Pfadangaben enthält.

PCT

Private Communication Technology. Ein Protokoll, das ähnlich wie → SSL eine sichere Verbindung ermöglicht. PCT ist mit SSL-Browsern kompatibel und verwendet längere und sicherere Schlüssel.

Private Communication Technology

Perl

Practical Extraction and Report Language. Eine Skriptsprache, die besonders für → CGI-Skripte benutzt wird. Perl wird als Interpreter realisiert.

Practical Extraction and Report Language

V
S
I
1
2
3
4
5
6
7
8
9
10
11

A

B

C

D

E

Pretty Good Privacy	PGP Pretty Good Privacy. Eine Anwendung zur Verschlüsselung von E-Mail, die weite Verbreitung gefunden hat.
	PHP Personal Homepage Tool oder auch PHP Hypertext Preprocessor. Ursprünglich von Rasmus Lerdorf entwickelte Skriptsprache für die Programmierung dynamischer Webseiten. PHP zeichnet sich durch großen Funktionsumfang und Plattformunabhängigkeit aus. So werden viele Datenbanken direkt unterstützt.
	PING <u>P</u> acket <u>I</u> nternet <u>G</u> opher. Ein Kommando, mit dem die Verbindung zu einem Host kontrolliert werden kann. Ping verwendet das ICMP Echo-Kommando.
	Portnummer Eine Nummer, die eine bestimmte Internetanwendung identifiziert. So läuft beispielsweise der WWW-Dienst des Internet Information Server auf Port 80.
Packet Internet Gopher	POST Methode zur Datenübertragung bei → HTTP.
	PostgresSQL Schlankes Datenbanksystem.
	PPP Point-to-Point Protocol. Ein Standard für den Aufbau von Verbindungen zu entfernten Computern.
Point-to-Point Protocol	PPTP Point-to-Point Tunneling Protocol. Ein Standard für die sichere Verbindung zwischen Computern und privaten Netzwerken. Über PPTP können verschiedene niedrigere Protokolle gefahren werden, darunter IP, IPX und NetBEUI.
Point-to-Point Tunneling Protocol	Prozess Unter Windows NT ein Objekt, bestehend aus einem ausführbaren Programm, einem Satz virtueller Speicheradressen und einem Thread (Ablaufplan). Unter UNIX ein Synonym für → Thread.
	Prozessisolation Ablauf eines Prozesses, einer Applikation oder Komponente außerhalb des Serverprozesses.
	Programmdatei Eine Datei, die ein ausführbares Programm ist oder eines startet.

Protokoll

Eine Methode, mit der Computer miteinander kommunizieren; beispielsweise beschreibt das Protokoll HTTP die Kommunikation im World Wide Web.

Proxy

Eine Software, die Nutzer mit einer entfernten Datenquelle über einen Zwischenspeicher verbindet. Ist der Speicher mit aktuellen Daten gefüllt, wird bei einem mehrfachen Zugriff nicht immer wieder die entfernte Verbindung benutzt.

Proxyserver

Ein Server, der als → Proxy konfiguriert wurde. Kann über erweiterte Funktionen wie beispielsweise eine → Firewall verfügen.

Q**Query**

Allgemein eine Abfrage, Anfrage oder Anforderung.

Query-Form

Ein Online-Formular, das der Nutzer ausfüllt und mit dem er Informationen anfordert, beispielsweise von einer Suchmaschine.

R**RAID**

Redundant Array of Inexpensive Disks. Eine Methode zur Bildung fehlertoleranter Systeme durch Anordnung von Festplatten in bestimmten Kombinationen. Es gibt sechs Arten von RAID (0 bis 5).

Redundant Array of Inexpensive Disks

RAM

Random Access Memory. Der Speicher, der im Computer als Schreib-Lese-Speicher arbeitet. Wird auch Arbeitsspeicher genannt. Bei Windows werden mindestens 128 MByte empfohlen, für größere Server bis 2 GByte. Für Linux reichen 64 MByte, als Server sind 512 MByte zu empfehlen.

Random Access Memory

RAS

Remote Access Service. Ein Dienst, der entfernten Computern den Zugriff über ein Netzwerk oder Modem erlaubt.

Remote Access Service

Redirection

Ein Prozess der automatischen Weiterleitung einer Anfrage eines Browsers zu einer anderen Adresse.

Registry, Registrierung

Eine Datenbank, die Bestandteil des Betriebssystems Windows ist, in der spezifische Informationen über das Betriebssystem und installierte Programme gespeichert sind. Die Registry ist hierarchisch strukturiert und enthält Einträge der Art Schlüssel:Wert.

V

S

I

1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E

Relationale Datenbank

Datenbanken nach diesem Modell speichern Daten in Tabellen, die in Spalten und Zeilen organisiert sind. Durch Relationen (Verbindungen) werden die Daten in Abhängigkeit voneinander organisiert.

Remote Administration

Administration eines Computers über ein Netzwerk von einem entfernten Computer aus.

Remote Data Services

Eine Technologie des Datenzugriffs über das Web mit ActiveX-Komponenten.

Replikation

Übertragung von Daten oder → Metadaten von einem Computer zu einem anderen. Die Replikation kann automatisch oder manuell erfolgen und dient dem Abgleich von Datenbanken; sie wird bei fehlertoleranten Systemen wie beispielsweise Clustern verwendet.

RFC**Request for Comments**

Request for Comments. Eine Serie von Dokumenten, mit der seit 1969 Protokolle und Techniken des Internets beschrieben werden. Die meisten Dokumente beschreiben nur bestimmte Zwischenschritte und Vorschläge. Aber alle Standards sind in RFCs zu finden. Die Verteilung erfolgt durch Organisationen wie ANSI. Die Vorstufe zum RFC ist das Draft.

Robot

Ein automatisch ablaufendes Programm, das Webserver durchsucht und die gefundenen Inhalte einer Suchmaschine zum Zweck der Indizierung zuleitet. Andere übliche Begriffe sind Spider oder Crawler.

Router

Ein Gerät in einem Netzwerk, das für die Herstellung von Verbindungen zwischen unterschiedlichen Netzwerken zuständig ist. Router können selbstständig unter verschiedenen Verbindungswegen wählen, wenn diese zur Verfügung stehen. Variable Wege können die Verbindungssicherheit entscheidend erhöhen. Alle Weiterschaltungen von Verbindungen im Internet laufen über Router.

RPC**Remote Procedure Call**

Remote Procedure Call. Ein Standard, der von der Open Software Foundation (OSF) definiert wurde. RPD erlaubt es einem Prozess, Funktionen eines anderen, ebenfalls gerade laufenden Prozesses, zu nutzen. Der Standort des Computers, auf dem der andere Prozess läuft, spielt dabei keine Rolle.

RSA

Ein Verschlüsselungsalgorithmus mit öffentlichen Schlüsseln, der im Internet verbreitet ist. Die Abkürzung ist ein Akronym aus den Anfangsbuchstaben der drei Erfinder Rivest, Shamir und Adleman.

run time

Laufzeit. Der Zeitpunkt, zu dem eine Applikation real abläuft, die Entwicklungsphase also beendet ist.

S**scope**

Sichtbereich. Definiert die Sicht oder Reichweite einer Variablen, einer Prozedur oder eines Objekts. Eine mögliche Sicht einer Variablen ist »public« (deutsch öffentlich). Die Variable ist dann in allen Prozeduren eines Skripts verfügbar.

Script, Skript

Eine Art Programm, das Befehle und Kommandos enthält und von einem Interpreter ausgeführt wird. Skripten sind oft einfacher als Programmiersprachen und verfügen über weniger Restriktionen. Sie sind oft deutlich langsamer in der Ausführung.

SGML

Standard Generalized Markup Language. Sprache zur Beschreibung von Dokumenten, Mutterdialekt für → HTML und → XML.

Skalierbarkeit

Im weitesten Sinn die Möglichkeit, eine Softwareumgebung unverändert auf einer ganzen Reihe unterschiedlicher Computer anzuwenden. Damit wird auch die Möglichkeit beschrieben, ohne grundlegende Eingriffe in die Software steigenden Anforderungen an den Datendurchsatz allein durch Änderungen der Hardware zu genügen. Die Software ist skalierbar.

Skript-Engine

Ein Programm, das Skripte interpretiert und ausführt. Die PHP-Engine ist ein solches Programm.

SDK

Software Development Kit. Ein Softwareentwicklungskit, das alle Programme und Werkzeuge enthält, um für ein bestimmtes System neue Programme und Applikationen zu entwickeln.

Semaphore

Ein Verriegelungsmechanismus innerhalb eines Ressource-Managers, der als Basismechanismus zum verteilten Zugriff auf Objekte benutzt wird.

V

S

I

1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E

Server

Ein allgemeiner Begriff für einen Computer, der in einem Netzwerk Daten, Dienste, Ressourcen oder Zugriff auf angeschlossene Daten, Dienste oder Ressourcen ermöglicht. Zu den Aufgaben eines Servers kann die Nutzerautorisierung und die Verwaltung von Sicherheitssystemen zählen. Beispiele: Applikationsserver führen Anwendungsprogramme aus, Daten- und Druckserver stellen nur Verzeichnis- und Druckdienste bereit, Webserver liefern Inhalte für das World Wide Web, Datenbankserver führen SQL-Datenbanken aus.

Serverzertifikate

Einmalige und eindeutige digitale Dokumente, die als Basis für verschlüsselte Verbindungen mit einem Webserver benutzt werden.

Server-Cluster

Eine Gruppe Computer, die in einem Netzwerk miteinander physisch und durch spezielle Software verbunden sind. Ein Cluster kann den Ausfall eines Teils der verbundenen Computer tolerieren; die anderen Computer übernehmen die Aufgabe der ausgefallenen Maschinen.

Server Node

Ein einzelner Computer in einem → Server-Cluster.

SGML

**Standard
Generalized
Markup Language**

Standard Generalized Markup Language. Ein ISO-Standard (ISO 8879:1986), der eine Metasprache zur Dokumentenformatierung beschreibt. Der Schwerpunkt liegt in der Strukturierung von Dokumenten für die elektronische Distribution. → HTML ist ein primitives Derivat von SGML.

SMTP

**Simple Mail
Transfer Protocol**

Simple Mail Transfer Protocol. Ein Protokoll, das in STD 10, RFC 821 definiert wurde und die Übertragung von E-Mail zwischen Servern im Internet beschreibt.

SNA

**Systems Network
Architecture**

Systems Network Architecture. Ein weitverbreitetes System aus Funktionen und Protokollen, das den Zugriff auf IBM-Computersysteme definiert.

SNMP

**Simple Network
Management
Protocol**

Simple Network Management Protocol. SNMP (RFC 1157) ist der Standard für die entfernte Kontrolle und Steuerung von Computern, Routern und anderen aktiven Netzwerkbauteilen. SNMP setzt auf TCP/IP auf.

Socket

Eine Software, welche die Verbindung zwischen einem Client und einem Server herstellt.

Spider

Ein anderer Begriff für → Robot.

spoofing

Schwindeln. Begriff für das Fälschen der eigenen Identität, meist durch Angabe einer falschen E-Mail-Adresse, URL, IP-Adresse usw.

SQL

Structured Query Language. Ein internationaler Standard für die Abfrage von relationalen Datenbanken.

Structured Query Language

SSI

Server Side Include. Datei oder Objekt, das in eine Webseite eingebunden und zum Webserver gesendet wird. Beispiele sind Kopf- oder Fußzeilen, die vor und nach jeder Webseite eingebunden werden.

Server Side Include

SSL

Secure Sockets Layer. Ein Protokoll, das die sichere, verschlüsselte Datenkommunikation ermöglicht. SSL verwendet den → RSA-Algorithmus für bestimmte TCP/IP-Ports.

Secure Sockets Layer

Statische Seite

Eine HTML-Seite, die fest auf einem Server abgelegt wird und nicht durch Reaktionen des Nutzers geändert werden kann. Im Gegensatz dazu werden dynamische Seiten erst auf Anforderung und dann von aktuellen Daten abhängig erzeugt.

Stub

Ein schnittstellenabhängiges Objekt, das auf Rufe eines Proxy-Objekts reagiert und die Verbindung zwischen zwei entfernten Prozessen herstellt. Der Stub befindet sich im Applikationsobjekt, der Proxy im Client.

Subnet Mask

Ein Konfigurationsparameter in einem TCP/IP-Netzwerk, der zur Trennung einer IP-Adresse von der Angabe des gesamten Netzwerks dient. Durch die Subnet Mask kann das »Subnetz« ermittelt werden.

SyBase

Professionelle Datenbank, die in mehreren Varianten verfügbar ist.

System Data Source Name (DSN)

Ein Name für eine → ODBC-Datenquelle, siehe auch → DSN.

T**T1**

In den USA eine 1,544-Mbps-Verbindung, in Europa 2,048 MBps.

T3

In den USA eine 45-Mbps-Verbindung (entspricht 28 T1).

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A

B
C
D
E

**Transmission
Control
Protocol/Internet
Protocol****TCP/IP**

Transmission Control Protocol/Internet Protocol. Ein Kommunikationsstandard für Netzwerke, auf dem das Internet basiert, der aber auch für lokale Netzwerke verwendet wird. Besteht eigentlich aus zwei Protokollen auf unterschiedlichen Schichten des OSI/ISO-Referenzmodells: TCP und IP. TCP splittet die Daten in Datensegmente. IP sammelt diese Segmente und sendet die Sammlungen, zusammen mit der Absender- und Empfängeradresse, als Pakete.

Telnet

Ein Protokoll, mit dem man auf einen entfernten Computer interaktiv so verbunden werden kann, dass die entfernte Konsole uneingeschränkt zur Verfügung steht.

Thread

Faden. Ein Abschnitt ausgeführter Software, die die CPU eines Computers zu einem bestimmten Zeitpunkt abarbeitet. Ein Thread kann Bestandteil einer Applikation sein. Alle Threads teilen sich den virtuellen Adressraum, globale Variablen und die Ressourcen eines Computersystems.

three-tier architecture

Drei-Schicht-Architektur. Teilt eine für Netzwerke entwickelte Applikation in die drei Schichten Nutzerschnittstelle, Anwendungsprogramm und Datenzugriff.

Timeout

Die Beendigung eines Vorgangs oder Prozesses aufgrund Zeitüberschreitung. Nach der Anforderung einer Antwort wird die Aktion auch bei Ausbleiben der Antwort abgebrochen, wenn die als Timeout festgelegte Zeitspanne überschritten wurde.

Transaktion

Eine Gruppe von Prozessen, die als eine Einheit ausgeführt werden müssen. Wenn die Einheit aufgrund einer technischen Störung nicht komplett ausgeführt werden kann, wird die gesamte Einheit zurück abgewickelt; die Änderungen werden storniert.

U**Universal Naming
Convention****UNC**

Universal Naming Convention. Eine Namenskonvention für die Angabe physischer Pfade; die Namen der betroffenen Server und Verzeichnisse werden mit einbezogen. Aliase werden nicht verwendet.

Unix

Betriebssystem mit präemptivem Multitasking, Multithreading und einer breiten Netzwerkfunktionalität. Wird vor allem im Serverbereich eingesetzt, wo es auf Leistung und Stabilität ankommt, weniger auf eine einfache Nutzerschnittstelle.

URL

Uniform Resource Locator. Eine Namenskonvention, die einen Ort auf einem Computer beschreibt. URLs können auch Aliase beinhalten. Die URL enthält außerdem Angaben über das Protokoll, mit dem auf den Ort zugegriffen werden kann, beispielsweise FTP oder HTTP.

Uniform Resource Locator

Usenet

Eine Hierarchie von Nachrichtengruppen (Newsgroups) im Internet.

V**VBA**

Microsoft VisualBasic for Applications. Eine Entwicklungsumgebung für die Entwicklung von Skripten für Anwendungsprogramme auf der Basis der Programmiersprache VisualBasic.

Microsoft VisualBasic for Applications

VBScript

Microsoft VisualBasic Scripting Edition. Ein Derivat der Sprache VisualBasic, die zur Programmierung von Webseiten im Webserver, für die Entwicklung von ActiveX-Controls im Browser und für Applets benutzt wird.

Microsoft VisualBasic Scripting Edition

Vererbung

Die Übertragung aller Eigenschaften einer Klasse auf eine neu zu bildende Klasse in einem objektorientierten Modell. Die abgeleitete Klasse kann um weitere Eigenschaften und Methoden ergänzt werden und verfügt dann über eine größere Funktionalität. Die Vererbung erleichtert die Wartung und Pflege komplexer Programme, die über Objekthierarchien verwaltet werden.

Virtuelles Verzeichnis

Ein Verzeichnis, das unter diesem Namen nicht physisch existiert, sondern auf ein anderes Verzeichnis abbildet.

VM

Virtuelle Maschine. Eine Software, die Java-Applets (Java Bytecode) interpretiert und ausführt und auf verschiedenen Computern zur Verfügung steht. Durch die VM wird Java plattformunabhängig.

Virtuelle Maschine

Virtueller Server

Wenn ein Computer mehrere Webs verwaltet und diese nach außen als eigenständige Einheiten erscheinen, wird von einem virtuellen Server gesprochen.

VRML

Virtual Reality Modeling Language. Eine Erweiterung der Sprache → HTML um Befehle zur Darstellung dreidimensionaler Objekte.

Virtual Reality Modeling Language

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

W

- World Wide Web Consortium** **W3C**
World Wide Web Consortium. Gegründet 1994, um weitere Standards für das World Wide Web zu entwickeln und zu propagieren. Das W3C befindet sich derzeit am Massachusetts Institute of Technology Laboratory for Computer Science (MIT/LCS), (USA), am Institut National de Recherche en Informatique et en Automatique (INRIA) (unter anderem in Le Chesnay Cedex, Frankreich) und an der Keio University Shonan Fujisawa Campus (Fujisawa, Japan). Das W3C wurde mit gegründet vom CERN (wo das WWW entwickelt wurde) und wird von der DARPA und der Europäischen Kommission unterstützt. Adresse: <http://www.w3.org/>.
- Wide Area Information Service** **WAIS**
Wide Area Information Service. Eine Methode für die Suche nach Informationen und für die Datenbankrecherche im Internet zugänglichen Datenbanken.
- Web Distributed Data Exchange** **WDDX**
Web Distributed Data Exchange. Eine → XML-Anwendung, die den Datenaustausch zwischen Webserver revolutionieren kann.
- Web-Applikation**
Eine Software, die → HTTP als Transportprotokoll benutzt und Informationen an Nutzer im → HTML-Format versendet.
- Webseite**
Ein Dokument, das im World Wide Web veröffentlicht wird. Webseiten können Texte, Bilder, Videos, Sound und Applets beinhalten.
- Webserver**
Ein → Server, der die Protokolle FTP und HTTP benutzt, um Daten und Informationen im Internet über → TCP/IP zur Verfügung zu stellen.
- World Wide Web** **WWW**
World Wide Web, oft nur als Web bezeichnet. Ein grafisch gestützter Dienst des Internet. Das WWW unterstützt mehrere Datenformate und die → Hypertextarchitektur. Das WWW ist ein Oberbegriff für alle → Webserver und Clients, die im Internet zusammengeschlossen sind und den Dokumentenaustausch auf der Basis des Protokolls → HTTP und der Dokumentenbeschreibungssprache → HTML vornehmen.

X

- eXtensible Markup Language** **XML**
eXtensible Markup Language. Ein Datenformat für strukturierte Dokumente mit einem flexiblen und anwendungsspezifisch erweiterba-

ren Sprachschatz. Durch XML kann die Dokumentenstruktur völlig von der Gestaltung getrennt werden.

XSL

Extensible Stylesheet Language. Formatierungssprache für XML-Daten, mit der die Darstellung der Informationen aus XML-Dateien gesteuert werden kann.

XSLT

→ XSL Transformation. Sprache mit Transformationsanweisungen, die eine Form von → XML in eine andere überführt. Damit sind innerhalb der XML-Welt Übertragungen von Daten möglich, ohne auf proprietäre Schnittstellen ausweichen zu müssen.

V

S

I

1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E

B Kurzreferenz

B.1 Vorbemerkungen

Diese Kurzreferenz soll bei der praktischen Arbeit mit PHP 4 helfen. Wenn Sie einen Funktionsnamen suchen oder nur schnell die Syntax nachschlagen wollen, wird dies eine wertvolle Hilfe sein. Wenn Sie noch weitergehende Hilfe benötigen, dann gibt es in Ergänzung dieses Buches die beste und umfangreichste deutschsprachige Referenz jetzt auch in gedruckter Form:

- **PHP 4. Die Referenz**, Carl Hanser Verlag, ISBN 3-446-21687-1

Schreibweisen in der Funktionsreferenz

Die Referenz stützt sich auf die übliche Schreibweise bei der Syntaxangabe. Da PHP überwiegend aus Funktionen besteht, wird die Funktionsschreibweise primär zu finden sein. Der folgende Ausdruck beschreibt das Syntaxdiagramm:

```
int function(string param1, 0|1 [, int param2])
```

Diese Schreibweise ist folgendermaßen zu interpretieren:

- **function()**. Der Funktionsname ist **fett** hervorgehoben. Wenn die Klammern zwingend erforderlich sind, gehören sie zum Funktionskopf und sind ebenso hervorgehoben.
- **int**. Vor der Funktion steht der Datentyp, den die Funktion zurückgibt. Im Beispiel gibt die Funktion also ganzzahlige Werte zurück.
- **string *param1***. Dieser Parameter ist nicht optional. Erwartet wird eine Zeichenkette. In der Erklärung zur Funktion wird auf *param1* verwiesen.
- **0|1**. Diese Parameter sind ebenso nicht optional, werden aber nicht durch einen Wert ersetzt, sondern so wie angegeben eingesetzt. Das |-Zeichen steht für eine Alternative, Sie können also entweder 0 oder 1 schreiben.
- **[int *param2*]**. Dieser Parameter wird als Ganzzahl (Integer) erwartet, ist aber optional – dies zeigen die eckigen Klammern an.

Der Aufruf der Funktion könnte im Skript so aussehen:

```
$intvar = function("test", 1, 45);
```

An einigen Stellen finden Sie das Wort **void**. Dies steht für »nichts«. Manchmal sind die Parameter auch untereinander (zeilenweise) angeordnet. Dies hat keine Bedeutung für die Funktion selbst. Es dient nur

der besseren Lesbarkeit. Schreiben Sie beim Programmieren immer alles in eine Zeile.

Was nicht aufgeführt wurde

CVS-Versionen wurden in dieser Übersicht nicht beachtet, da diese Entwicklerstadien den meisten Lesern nicht zugänglich sein dürften. Keine Aufnahme fanden auch Funktionen, die nur einige Versionen lang verfügbar waren und dann wieder verschwanden. Insbesondere sind solche nicht aufgeführt, die den Sprung von PHP 3 nach PHP 4 nicht überlebt haben. Sie sollten diese Funktionen nicht verwenden, auch wenn sie in anderen Quellen oder in der Dokumentation stehen. Einige Funktionen sind auch in der offiziellen PHP-Dokumentation nicht aufgeführt. Alle Funktionen sind jedoch in der alphabetischen Liste zu finden. Steht keine Seitenangabe dahinter, erscheint sie nicht in der Referenz.

**Wo Sie das
offizielle Manual
finden**

Einige selten benötigte Funktionen fanden aus Platzgründen keine Aufnahme. Hier sei auf die zur Ergänzung geeignete Dokumentation der PHP Documentations Group verwiesen, die Sie unter der folgenden Adresse finden:

<http://www.php.net/manual>

B.2 Alphabetische Übersicht

Die folgende Übersicht listet die Funktionsnamen in PHP in alphabetischer Reihenfolge auf. Die Liste bezieht sich auf Version 4.0.6. Alle Funktionsnamen sind zugleich reservierte Namen. Vermeiden Sie die Verwendung reservierter Namen für Variablen und Konstanten. Da nicht alle Funktionen dokumentiert sind und einige ausgesprochen selten benötigt werden, sind nicht immer Querverweise in die Referenz zu finden.

Zu jeder Funktion ist außerdem die Version aufgeführt, mit der sie eingeführt wurde. Funktionen, die bei früheren Versionen vorhanden waren, später aber nicht mehr auftauchten und damit heute nicht zur Verfügung stehen, wurden ebenfalls nicht mit aufgenommen.

Schreibweisen in der alphabetischen Übersicht

Achten Sie auf folgende Schreibweisen in der Liste:

Ziel des Verweises

- `functionsname.....265, [3.15]`. Die Funktion wird auf Seite 265 in der Referenz beschrieben, sie ist seit PHP 3.0.15 verfügbar. Entsprechend steht 3.00 für 3.0.0, 3.01 für 3.0.1 und 3.10 für 3.0.10. Es gab bis Mitte 2000, dem endgültigen Erscheinungster-

min von PHP 4, insgesamt 18 Releases von PHP 3.0: 3.0.0 bis 3.0.17. Weitere werden trotz PHP 4 folgen.

- `functions_name` ([4.b2]). Die Funktion wird nicht beschrieben, sie ist noch nicht dokumentiert. Sie ist mit PHP 4, 2. Beta, eingeführt worden (es gab drei öffentliche Betaversionen: 1, 2 und 4). 4RC1 steht für Release Candidate 1, RC 2 entsprechend für Release Candidate 2 (es gab zwei RCs). 4.0.0, 4.0.1, 4.0.2 usw. stehen für die offiziellen Final-Releases, dieses Buch geht teilweise bis zur Version PHP 4.0.6 (Stand: September 2001). **Version**
- `functions_name` ([3.13][4RC1]). Die Funktion ist seit 3.0.13 verfügbar, war jedoch in den ersten Betaversionen von PHP 4 nicht mehr vertreten und tauchte im Release Candidate 1 wieder auf. Sie wäre also beispielsweise in PHP 3.0.16 verfügbar, in PHP 4 Beta2 jedoch nicht.
- `functions_name` (*aliasname*). Der Funktionsname ist unter dem Alias *aliasname* gebräuchlicher, sehen Sie dort nach. **Aliase**

```

pdf_set_horiz_scaling ..... 335, [3.06]
pdf_set_info..... 325, [4.01]
pdf_set_info_author (pdf_set_info)
pdf_set_info_creator (pdf_set_info)
pdf_set_info_keywords (pdf_set_info)

```

Abbildung B.1:
Bei Funktionsnamen,
die nicht mehr im
Gebrauch sind, wird
auf den neuen
Namen verwiesen

Behandlung der Sprachkonstrukte

Einige »Funktionsnamen« sind **fett** geschrieben, beispielsweise **array** oder **die**. Dies sind keine Funktionen, sondern Sprachkonstrukte oder Anweisungen, die wie Funktionen aufgerufen werden. Da viele Leser solche Konstrukte vermutlich in der Liste suchen, sind sie trotzdem dort mit aufgenommen worden. Um korrekt zu bleiben, sind sie aber besonders gekennzeichnet.

**Nicht alles sind
Funktionen**

A

abs 949, [3.00]
 acos 949, [3.00]
 addslashes 929, [4.b4]
 addslashes 929, [3.00]
 apache_lookup_uri 1070, [3.04]
 apache_note 1070, [3.02]
array 922, [3.00]
 array_count_values 926, [4.b4]
 array_diff 926, [4.01]
 array_flip 926, [4.b4]
 array_intersect 926, [4.01]
 array_keys 926, [4.b1]
 array_merge 926, [4.b1]
 array_merge_recursive 926, [4.01]
 array_multisort 927, [4.b4]
 array_pad 926, [4.b4]
 array_pop 927, [4.b1]
 array_push 927, [4.b1]
 array_rand 928, [4.02]
 array_reverse 927, [4.b4]
 array_shift 927, [4.b1]
 array_slice 928, [4.b1]
 array_splice 928, [4.b1]
 array_sum 928, [4.04]
 array_unique 928, [4.01]
 array_unshift 928, [4.b1]
 array_values 928, [4.b1]
 array_walk 922, [3.03]
 arsort 922, [3.00]
 asin 949, [3.00]
 asort 922, [3.00]
 assert 1078, [4.b4]
 assert_options 1078, [4.b4]
 atan 949, [3.00]
 atan2 950, [3.05]

B

base64_decode 917, [3.00]
 base64_encode 917, [3.00]
 base_convert 950, [3.06]
 basename 956, [3.00]
 bcadd 954, [3.00]
 bccomp 954, [3.00]
 bcddiv 954, [3.00]
 bcmod 954, [3.00]
 bcmul 954, [3.00]
 bcpow 954, [3.00]
 bcscale 954, [3.00]
 bcsqrt 955, [3.00]
 bcsub 955, [3.00]
 bin2hex 929, [3.10]
 bindec 950, [3.00]
 bindtextdomain 942, [3.07]
break 908, [3.00]

C

call_user_func 915, [3.03]
 call_user_func_array 915, [4.04]
 call_user_method 912, [3.03]
 call_user_method_array 912, [4.04]
 ceil 950, [3.00]
 chdir 955, [3.00]
 checkdate 945, [3.00]
 checkdnsrr 971, [3.00]
 chgrp 956, [3.00]
 chmod 957, [3.00]

chop	930, [3.00]	curl_init.....	387, [4.02]
chown	956, [3.00]	curl_setopt.....	973, [4.02]
chr	930, [3.00]	curl_version.....	393, [4.02]
chunk_split	930, [3.06]	current.....	923, [3.08]
class	910, [3.00]	D	
class_exists	913, [4.01]	date.....	945, [3.00]
clearstatcache	957, [3.00]	dba_close.....	1032, [3.08]
closedir	955, [3.00]	dba_delete.....	1033, [3.08]
closelog	1073, [3.00]	dba_exists.....	1032, [3.08]
compact	929, [4.b1]	dba_fetch.....	1032, [3.08]
com	913, [4.b2]	dba_firstkey.....	1033, [3.08]
com_invoke	914, [3.03]	dba_insert.....	1032, [3.08]
com_load	914, [3.03]	dba_nextkey.....	1033, [3.08]
com_propget	914, [3.03]	dba_open.....	1032, [3.08]
com_propput	914, [3.03]	dba_optimize.....	1033, [3.08]
connection_aborted	1069, [3.07]	dba_popen.....	1032, [3.08]
connection_status	1069, [3.07]	dba_replace.....	1033, [3.08]
connection_timeout	1069, [3.07]	dba_sync.....	1033, [3.08]
constant	911, [4.04]	dbase_add_record.....	1035, [3.00]
continue	909, [3.00]	dbase_close.....	1035, [3.00]
convert_cyr_string	930, [3.06]	dbase_create.....	1035, [3.00]
copy	957, [3.00]	dbase_delete_record.....	1035, [3.00]
cos	950, [3.00]	dbase_get_record.....	1036, [3.00]
count	922, [3.00]	dbase_get_record_with_names..	1036, [3.05]
count_chars	930, [4.b4]	dbase_numfields.....	1036, [3.00]
create_function	915, [4.01]	dbase_numrecords.....	1036, [3.00]
crc32	930, [4.01]	dbase_open.....	1035, [3.00]
crypt	930, [3.08]	dbase_pack.....	1035, [3.00]
curl_close	972, [4.02]	dbase_replace_record.....	1035, [3.11]
curl_errno	972, [4.03]	dblist.....	1034, [3.00]
curl_error	972, [4.03]	dbmclose.....	1033, [3.00]
curl_exec	972, [4.02]	dbmdelete.....	1034, [3.00]
curl_getinfo	972, [4.04]	dbmexists.....	1034, [3.00]

- dbmfetch 1034, [3.00]
 - dbmfirstkey 1034, [3.00]
 - dbminsert 1034, [3.00]
 - dbmnextkey 1034, [3.00]
 - dbmopen 1033, [3.00]
 - dbmreplace 1034, [3.00]
 - dcgettext 942, [3.08]
 - debugger_off 1079, [3.00]
 - debugger_on 1078, [3.00]
 - decbin 950, [3.00]
 - dechex 950, [3.00]
 - decoct 950, [3.00]
 - define 911, [3.00]
 - defined 911, [3.00]
 - define_syslog_variables 1072, [3.00]
 - deg2rad 951, [3.04]
 - dgettext 942, [3.07]
 - delete 965, [3.00]
 - die** 1072, [3.00]
 - dir (*getdir*)
 - dirname 957, [3.00]
 - diskfreespace 957, [3.07]
 - d1 1071, [3.00]
 - domxml_add_root 985, [4.b4]
 - domxml_attributes 986, [4.b4]
 - domxml_attrname 986, [4.04]
 - domxml_children 986, [4.b4]
 - domxml_dtd 986, [4.04]
 - domxml_dumpmem 986, [4.b4]
 - domxml_getattr 986, [4.b4]
 - domxml_lastchild 986, [4.04]
 - domxml_new_child 986, [4.b4]
 - domxml_new_xml doc 987, [4.b4]
 - domxml_node 987, [4.b4]
 - domxml_parent 987, [4.04]
 - domxml_root 987, [4.b4]
 - domxml_rootnew 987, [4.04]
 - domxml_setattr 987, [4.b4]
 - domxml_set_content 987, [4.04]
 - doubleval 919, [3.00]
- ## E
- each 923, [3.00]
 - easter_date 949, [3.09]
 - easter_days 949, [3.09]
 - echo** 930, [3.00]
 - empty** 919, [3.00]
 - else** 907, [3.00]
 - elseif** 907, [3.00]
 - end 923, [3.00]
 - endif** (*if*)
 - ereg 943, [3.00]
 - ereg_replace 943, [3.00]
 - eregi 943, [3.00]
 - eregi_replace 943, [3.00]
 - error_log 1079, [3.00]
 - error_reporting 1079, [3.00]
 - escapeshellarg 1071, [4.03]
 - escapeshellcmd 1071, [3.00]
 - eval** 1072, [3.00]
 - exec 1071, [3.00]
 - exit** 1072, [3.00]
 - exp 951, [3.00]
 - explode 931, [3.00]
 - extends** 911, [3.00]
 - extension_loaded 1076, [3.10]
 - extract 929, [3.07]

F

fclose	957, [3.00]	fputs (<i>fwrite</i>)	960, [3.00]
feof	957, [3.00]	fread	960, [3.00]
fflush	958, [4.01]	frenchtojd	948, [3.00]
fgetc	958, [3.00]	fscanf	960, [4.01]
fgetcsw	958, [3.08]	fseek	960, [3.00]
fgets	958, [3.00]	fsockopen	970, [3.00]
fgetss	958, [3.00]	fstat	961, [4RC1]
file	958, [3.00]	ftell	961, [3.00]
file_exists	958, [3.00]	ftp_cdup	974, [3.13] [4.b4]
fileatime	959, [3.00]	ftp_chdir	974, [3.13] [4.b4]
filectime	959, [3.00]	ftp_connect	973, [3.13] [4.b4]
filegroup	959, [3.00]	ftp_delete	976, [3.13] [4.b4]
fileinode	959, [3.00]	ftp_exec	973, [3.13] [4.03]
filemtime	959, [3.00]	ftp_fget	975, [3.13] [4.b4]
fileowner	959, [3.00]	ftp_fput	975, [3.13] [4.b4]
fileperms	959, [3.00]	ftp_get	975, [3.13] [4.b4]
filepro	1036, [3.00]	ftp_login	973, [3.13] [4.b4]
filepro_fieldcount	1037, [3.00]	ftp_mdtm	975, [3.13] [4.b4]
filepro_fieldname	1036, [3.00]	ftp_mkdir	974, [3.13] [4.b4]
filepro_fieldtype	1036, [3.00]	ftp_nlist	974, [3.13] [4.b4]
filepro_fieldwidth	1037, [3.00]	ftp_pasv	975, [3.13] [4.b4]
filepro_retrieve	1037, [3.00]	ftp_put	975, [3.13] [4.b4]
filepro_rowcount	1037, [3.00]	ftp_pwd	973, [3.13] [4.b4]
filesize	959, [3.00]	ftp_quit	976, [3.13] [4.b4]
filetype	960, [3.00]	ftp_rawlist	974, [3.13] [4.b4]
flock	960, [3.07]	ftp_rename	976, [3.13] [4.b4]
floor	951, [3.00]	ftp_rmdir	974, [3.13] [4.b4]
flush	1068, [3.00]	ftp_site	974, [3.13] [4.b4]
for	908, [3.00]	ftp_size	975, [3.13] [4.b4]
foreach	908, [4.00]	ftp_systype	974, [3.13] [4.b4]
fopen	960, [3.00]	ftruncate	961, [4RC1]
fpass thru	960, [3.00]	function	910, [3.00]
		function_exists	1079, [3.07]

func_get_arg 914, [4RC1]
 func_get_args 914, [4RC1]
 func_num_args 914, [4RC1]
 fwrite 961, [3.00]

G

getallheaders 1070, [3.00]
 getcwd 956, [4.b4]
 getdate 946, [3.00]
 getenv 1073, [3.00]
 gethostbyaddr 971, [3.00]
 gethostbyname 971, [3.00]
 gethostbyname1 971, [3.00]
 getimagesize 1017, [3.00]
 getlastmod 1074, [3.00]
 getmxrr 971, [3.00]
 getmyinode 1074, [3.00]
 getmypid 1074, [3.00]
 getmyuid 1074, [3.00]
 getprotobyname 971, [4.b4]
 getprotobynumber 972, [4.b4]
 getrandmax 951, [3.00]
 getrusage 1074, [3.07]
 getservbyname 972, [4.b4]
 getservbyport 972, [4.b4]
 gettext 943, [3.07]
 gettimeofday 946, [3.07]
 gettype 918, [3.00]
 getallheaders 918, [4.04]
 get_browser 1077, [3.00]
 get_cfg_var 1073, [3.00]
 get_class 912, [4.00]
 get_class_methods 912, [4.00]
 get_class_vars 913, [4.00]
 get_current_user 1073, [3.00]
 get_declared_classes 913, [4.00]
 get_defined_functions 1077, [4.04]
 get_defined_vars 1077, [4.04]
 get_extension_funcs 1078, [4.b2]
 get_html_translation_table ... 931, [4.b2]
 get_included_files 1074, [4.01]
 get_loaded_extensions 1078, [4.b2]
 get_magic_quotes_gpc 1073, [3.06]
 get_magic_quotes_runtime 1074, [3.06]
 get_meta_tags 931, [3.06]
 get_object_vars 913, [4.00]
 get_parent_class 912, [4.00]
 get_required_files 1074, [4.01]
 get_resource_type 1078, [4.02]
 gmdate 946, [3.00]
 gmmktime 947, [3.00]
 gmstrftime 946, [3.12]
 gregoriantojd 947, [3.00]
 gzclose 965, [3.00]
 gzcompress 965, [4.02]
 gzeof 965, [3.00]
 gzfile 965, [3.00]
 gzgetc 966, [3.00]
 gzgets 966, [3.00]
 gzgetss 966, [3.00]
 gzopen 966, [3.00]
 gzpassthru 966, [3.00]
 gzputs (*gzwrite*)
 gzread 966, [3.00]
 gzrewind 966, [3.00]
 gzseek 967, [3.00]
 gztell 967, [3.00]

gzuncompress 967, [4.02]
 gzwrite 967, [3.00]

H

header 918, [3.00]
 headers_sent 918, [3.08]
 hexdec 951, [3.00]
 highlight_file 1074, [4.b1]
 highlight_string 1075, [4.b1]
 htmlentities 931, [3.00]
 htmlspecialchars 931, [3.00]
 hw_array2objrec 1025, [3.04]
 hw_children 1025, [3.03]
 hw_childrenobj 1025, [3.03]
 hw_close 1025, [3.03]
 hw_connect 1025, [3.03]
 hw_cp 1025, [3.03]
 hw_deleteobject 1025, [3.03]
 hw_docbyanchor 1026, [3.03]
 hw_docbyanchorobj 1026, [3.03]
 hw_document_attributes 1026, [3.03]
 hw_document_bodytag 1026, [3.03]
 hw_document_content 1026, [3.03]
 hw_document_setcontent 1026, [3.03]
 hw_document_size 1026, [3.03]
 hw_edittext 1027, [3.03]
 hw_error 1027, [3.03]
 hw_errormsg 1026, [3.03]
 hw_free_document 1027, [3.03]
 hw_getanchors 1029, [3.03]
 hw_getanchorsobj 1029, [3.03]
 hw_getandlock 1028, [3.03]
 hw_getchildcoll 1027, [3.03]
 hw_getchildcollobj 1027, [3.03]

hw_getchilddoccoll 1029, [3.03]
 hw_getchilddoccollobj 1029, [3.03]
 hw_getobject 1028, [3.03]
 hw_getobjectbyquery 1028, [3.03]
 hw_getobjectbyquerycoll 1029, [3.03]
 hw_getobjectbyquerycollobj 1029, [3.03]
 hw_getobjectbyqueryobj 1028, [3.03]
 hw_getparents 1027, [3.03]
 hw_getparentsobj 1027, [3.03]
 hw_getremote 1027, [3.03]
 hw_getremotechildren 1028, [3.03]
 hw_getsrcbydestobj 1028, [3.03]
 hw_gettext 1028, [3.03]
 hw_getusername 1032, [3.03]
 hw_identify 1030, [3.03]
 hw_incollections 1030, [3.03]
 hw_info 1030, [3.03]
 hw_inscoll 1030, [3.03]
 hw_insdock 1030, [3.03]
 hw_insertdocument 1030, [3.03]
 hw_insertobject 1030, [3.03]
 hw_modifyobject 1030, [3.03]
 hw_mv 1029, [3.03]
 hw_new_document 1031, [3.03]
 hw_objrec2array 1031, [3.03]
 hw_outputdocument 1031, [3.03]
 hw_pconnect 1031, [3.03]
 hw_pipedocument 1031, [3.03]
 hw_root 1031, [3.03]
 hw_unlock 1031, [3.03]
 hw_who 1031, [3.03]

I

if 907, [3.00]

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

ifx_affected_rows	1038, [3.00]	imagechar	1017, [3.00]
ifx_blobinfile_mode	1040, [3.00]	imagecharup	1017, [3.00]
ifx_byteasvarchar	1041, [3.00]	imagecolorallocate	1018, [3.00]
ifx_close	1037, [3.00]	imagecolorat	1022, [3.00]
ifx_connect	1037, [3.00]	imagecolorclosest	1023, [3.00]
ifx_copy_blob	1040, [3.02]	imagecolordeallocate	1018, [3.06]
ifx_create_blob	1040, [3.02]	imagecolorexact	1023, [3.00]
ifx_create_char	1039, [3.03]	imagecolorresolve	1023, [3.00]
ifx_do	1038, [3.01]	imagecolorset	1023, [3.00]
ifx_error	1038, [3.00]	imagecolorsforindex	1023, [3.00]
ifx_errormsg	1038, [3.01]	imagecolorstotal	1023, [3.00]
ifx_fetch_row	1038, [3.01]	imagecolortransparent	1018, [3.00]
ifx_fieldproperties	1039, [3.01]	imagecopy	1018, [3.06]
ifx_fieldtypes	1039, [3.01]	imagecopymerge	1018, [4.01]
ifx_free_blob	1040, [3.01]	imagecopymergegray	1018, [4.06]
ifx_free_char	1039, [3.04]	imagecopyresized	1018, [3.00]
ifx_free_result	1039, [3.01]	imagecreate	1019, [3.00]
ifx_get_blob	1040, [3.02]	imagecreatefromgif	1019, [3.00]
ifx_get_char	1040, [3.04]	imagecreatefromjpeg	1019, [3.16]
ifx_getsqlca	1038, [3.06]	imagecreatefrompng	1019, [3.13]
ifx_htmltbl_result	1038, [3.00]	imagecreatefromstring	1019, [4.04]
ifx_nullformat	1041, [3.01]	imagecreatefromwbmp	1019, [4.02]
ifx_num_fields	1039, [3.00]	imagedashedline	1019, [3.00]
ifx_num_rows	1039, [3.00]	imagedestroy	1019, [3.00]
ifx_pconnect	1037, [3.00]	imagefill	1020, [3.00]
ifx_prepare	1038, [3.01]	imagefilledpolygon	1020, [3.00]
ifx_query	1037, [3.03]	imagefilledrectangle	1020, [3.00]
ifx_textasvarchar	1040, [3.04]	imagefilltoborder	1020, [3.00]
ifx_update_blob	1040, [3.04]	imagefontheight	1020, [3.00]
ifx_update_char	1039, [3.06]	imagefontwidth	1020, [3.00]
ifxus_close_slob	1041, [3.01]	imagegif	1020, [3.00]
ignore_user_abort	1069, [3.07]	imagejpeg	1020, [3.16]
imagearc	1017, [3.00]	imageinterlace	1021, [3.00]

imageline	1021, [3.00]	imap_deletemailbox.....	1008, [3.00]
imageloadfont	1021, [3.00]	imap_errors.....	1008, [3.12]
imagepng	1021, [3.16]	imap_expunge.....	1008, [3.00]
imagepolygon	1021, [3.00]	imap_fetchbody.....	1008, [3.00]
imagepsbbox	1024, [3.08]	imap_fetchheader.....	1013, [3.05]
imagepsencodefont	1024, [3.08]	imap_fetchstructure.....	1009, [3.00]
imagepsextendfont	1024, [4RC1]	imap_fetchtext (<i>imap_body</i>)	
imagepsfreefont	1024, [3.08]	imap_fetch_overview.....	1009, [3.07]
imagepsloadfont	1023, [3.08]	imap_getmailboxes (<i>imap_list_full</i>)	
imagepslantfont	1024, [3.08]	imap_getsubscribed (<i>imap_lsub_full</i>)	
imagepstext	1024, [3.08]	imap_header.....	1009, [3.00]
imagerectangle	1021, [3.00]	imap_headerinfo (<i>imap_header</i>)	
imagesetpixel	1021, [3.00]	imap_headers.....	1009, [3.00]
imagestring	1022, [3.00]	imap_last_error.....	1009, [3.12]
imagestringup	1022, [3.00]	imap_list (<i>imap_listmailbox</i>)	
imagesx	1022, [3.00]	imap_listmailbox.....	1009, [3.00]
imagesy	1022, [3.00]	imap_listsubscribed.....	1009, [3.00]
imagetypes	1022, [4.02]	imap_lsub (<i>imap_listsubscribed</i>)	
imagettfbbox	1022, [3.01]	imap_mail.....	1010, [3.14]
imagettfbtext	1022, [3.00]	imap_mail_compose.....	1010, [3.06]
imagewbmp	1021, [3.14]	imap_mail_copy.....	1010, [3.00]
imap_8bit	1011, [3.00]	imap_mail_move.....	1010, [3.00]
imap_alerts	1007, [3.12]	imap_mailboxmsginfo.....	1012, [3.02]
imap_append	1007, [3.00]	imap_msgno.....	1012, [3.04]
imap_base64	1007, [3.00]	imap_num_msg.....	1010, [3.00]
imap_binary	1007, [3.02]	imap_num_recent.....	1010, [3.00]
imap_body	1007, [3.00]	imap_open.....	1010, [3.00]
imap_check	1007, [3.00]	imap_ping.....	1010, [3.00]
imap_clearflag_full	1007, [3.04]	imap_popen.....	1012, [3.12]
imap_close	1007, [3.00]	imap_qprint.....	1011, [3.00]
imap_create (<i>imap_createmailbox</i>)		imap_rename (<i>imap_renamemailbox</i>)	
imap_createmailbox	1008, [3.00]	imap_renamemailbox.....	1011, [3.00]
imap_delete	1008, [3.00]	imap_reopen.....	1011, [3.00]

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

imap_rfc822_parse_adrlist ... 1012, [3.02]	is_float (<i>is_double</i>)
imap_rfc822_parse_headers ... 1012, [4.b4]	is_int (<i>is_integer</i>)
imap_rfc822_write_address ... 1012, [3.02]	is_integer 920, [3.00]
imap_scan (<i>imap_scanmailbox</i>)	is_link 962, [3.00]
imap_scanmailbox 1012, [3.00]	is_long 920, [3.00]
imap_search 1013, [3.12]	is_null 920, [4.04]
imap_setflag_full 1012, [3.03]	is_numeric 920, [4.b4]
imap_sort 1013, [3.03]	is_object 920, [3.00]
imap_status 1013, [3.04]	is_readable 962, [3.00]
imap_subscribe 1013, [3.00]	is_resource 921, [4.b2]
imap_uid 1013, [3.03]	is_real (<i>is_double</i>)
imap_undelete 1013, [3.00]	is_scalar 921, [4.05]
imap_unsubscribe 1013, [3.00]	is_string 921, [3.00]
imap_utf7_decode 1014, [3.15]	is_subclass_of 913, [4.00]
imap_utf7_encode 1014, [3.15]	is_uploaded_file 962, [4.03]
imap_utf8 1014, [3.13]	is_writeable 962, [3.00]
implode 931, [3.00]	isset 921, [3.00]
ini_alter (<i>ini_set</i>)	
ini_get 1075, [4.b1]	
ini_restore 1075, [4.b1]	
ini_set 1075, [4.b1]	
intval 919, [3.00]	
in_array 929, [4.b1]	
ip2long 970, [4.b4]	
include 910, [3.00]	
include_once 910, [4.b1]	
iptcparse 1024, [3.07]	
is_array 919, [3.00]	
is_bool 919, [4.b4]	
is_dir 961, [3.00]	
is_double 919, [3.00]	
is_executable 961, [3.00]	
is_file 962, [3.00]	
	J
	jddayofweek 949, [3.00]
	jdmonthname 948, [3.00]
	jdtofrrench 948, [3.00]
	jdtogregorian 947, [3.00]
	jdtojewish 948, [3.00]
	jdtojulian 947, [3.00]
	jdtounix 948, [4RC2]
	jewishtojd 948, [3.00]
	join 931, [3.00]
	juliantojd 948, [3.00]
	K
	key 923, [3.00]
	krsort 923, [3.13]
	ksort 923, [3.00]

L

ldap_add 1014, [3.00]
 ldap_bind 1014, [3.00]
 ldap_close 1015, [3.00]
 ldap_connect 1015, [3.00]
 ldap_count_entries 1015, [3.00]
 ldap_delete 1015, [3.00]
 ldap_dn2ufn 1015, [3.00]
 ldap_explode_dn 1015, [3.00]
 ldap_first_attribute 1015, [3.00]
 ldap_first_entry 1015, [3.00]
 ldap_free_result 1016, [3.00]
 ldap_get_attributes 1016, [3.00]
 ldap_get_dn 1016, [3.00]
 ldap_get_entries 1016, [3.00]
 ldap_get_values 1016, [3.00]
 ldap_get_values_len 1016, [4.00]
 ldap_list 1016, [3.00]
 ldap_mod_add 1014, [3.00]
 ldap_mod_del 1014, [3.00]
 ldap_mod_replace 1014, [3.00]
 ldap_modify 1016, [3.00]
 ldap_next_attribute 1016, [3.00]
 ldap_next_entry 1017, [3.00]
 ldap_read 1017, [3.00]
 ldap_search 1017, [3.00]
 ldap_unbind 1017, [3.00]
 leak 1071, [3.00]
 levenshtein 932, [4.01]
 link 962, [3.00]
 linkinfo 962, [3.00]
 list 923, [3.00]
 localtime 946, [4.00]

log 951, [3.00]
 log10 951, [3.00]
 long2ip 970, [4.b4]
 lstat 964, [3.04]
 ltrim 932, [3.00]

M

mail 1007, [3.00]
 magic_quotes_runtime
 (*set_magic_quotes_runtime*)
 max 952, [3.00]
 mcrypt_cbc 977, [3.09]
 mcrypt_cfb 977, [3.09]
 mcrypt_create_iv 976, [3.09]
 mcrypt_ecb 977, [3.09]
 mcrypt_get_block_size 976, [3.09]
 mcrypt_get_cipher_name 976, [3.09]
 mcrypt_get_key_size 976, [3.09]
 mcrypt_ofb 977, [3.09]
 mcrypt_list_algorithms 977, [4.02]
 mcrypt_list_modes 977, [4.02]
 mcrypt_get_iv_size 977, [4.02]
 mcrypt_encrypt 977, [4.02]
 mcrypt_decrypt 978, [4.02]
 mcrypt_module_open 978, [4.02]
 mcrypt_generic_init 978, [4.02]
 mcrypt_generic 978, [4.02]
 mdcrypt_generic 978, [4.02]
 mcrypt_generic_end 978, [4.02]
 mcrypt_enc_self_test 978, [4.02]
 mcrypt_enc_is_block_algorithm_mode
 979, [4.02]
 mcrypt_enc_is_block_algorithm 979, [4.02]
 mcrypt_enc_is_block_mode 979, [4.02]

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

mcrypt_enc_get_block_size 979, [4.02]	move_uploaded_file 963, [4.03]
mcrypt_enc_get_key_size 979, [4.02]	mysql (<i>mysql_db_query</i>)
mcrypt_enc_get_supported_key_sizes 979, [4.02]	mysql_affected_rows 1045, [3.00]
mcrypt_enc_get_iv_size 979, [4.02]	mysql_close 1045, [3.00]
mcrypt_enc_get_algorithms_name 980, [4.02]	mysql_connect 1045, [3.00]
mcrypt_enc_get_modes_name 980, [4.02]	mysql_create_db 1045, [3.00]
mcrypt_module_self_test 980, [4.02]	mysql_data_seek 1045, [3.00]
mcrypt_module_is_block_algorithm_mode 980, [4.02]	mysql_dbname 1046, [3.00]
mcrypt_module_is_block_algorithm 980, [4.02]	mysql_db_query ([3.00])
mcrypt_module_is_block_mode .. 980, [4.02]	mysql_drop_db 1046, [3.00]
mcrypt_module_get_algo_block_size 980, [4.02]	mysql_error 1046, [3.00]
mcrypt_module_get_algo_key_size 981, [4.02]	mysql_fetch_array 1046, [3.00]
mcrypt_module_get_algo_ supported_key_sizes 981, [4.02]	mysql_fetch_field 1046, [3.00]
md5 932, [3.00]	mysql_fetch_object 1046, [3.00]
mdecrypt_generic 978, [4.02]	mysql_fetch_row 1046, [3.00]
metaphone 932, [4.b4]	mysql_field_seek 1047, [3.00]
method_exists 913, [4.00]	mysql_field_flags 1047, [3.07]
mhash 981, [3.09]	mysql_field_len 1047, [3.07]
mhash_count 981, [3.09]	mysql_field_name 1046, [3.07]
mhash_get_block_size 981, [3.09]	mysql_field_table 1047, [3.07]
mhash_get_hash_name 981, [3.09]	mysql_field_type 1047, [3.07]
mhash_keygen_s2k 981, [4.04]	mysql_free_result 1047, [3.00]
microtime 947, [3.00]	mysql_list_dbs 1047, [3.00]
min 952, [3.00]	mysql_list_fields 1047, [3.00]
mkdir 962, [3.00]	mysql_list_tables 1048, [3.00]
mktime 946, [3.00]	mysql_num_fields 1048, [3.00]
	mysql_num_rows 1048, [3.00]
	mysql_pconnect 1048, [3.00]
	mysql_query 1048, [3.00]
	mysql_regcase 1048, [3.00]
	mysql_result 1048, [3.00]
	mysql_select_db 1048, [3.00]
	mysql_tablename 1049, [3.00]

mssql_affected_rows	1042, [3.06]	mysql_drop_db.....	1050, [3.00]
mssql_close	1042, [3.00]	mysql_errno.....	1050, [3.00]
mssql_connect	1042, [3.00]	mysql_error.....	1050, [3.00]
mssql_data_seek	1042, [3.00]	mysql_escape_string.....	1050, [4.03]
mssql_fetch_array	1042, [3.00]	mysql_fetch_array.....	1050, [3.00]
mssql_fetch_field	1043, [3.00]	mysql_fetch_assoc.....	1050, [4.03]
mssql_fetch_object	1043, [3.00]	mysql_fetch_field.....	1051, [3.00]
mssql_fetch_row	1043, [3.00]	mysql_fetch_lengths.....	1051, [3.00]
mssql_field_length	1043, [4.b2]	mysql_fetch_object.....	1051, [3.00]
mssql_field_name	1043, [4.b2]	mysql_fetch_row.....	1051, [3.00]
mssql_field_seek	1043, [3.00]	mysql_field_flags.....	1052, [3.00]
mssql_field_type	1043, [4.b2]	mysql_field_len.....	1052, [3.00]
mssql_free_result	1043, [3.00]	mysql_field_name.....	1051, [3.00]
mssql_get_last_message	1044, [4.00]	mysql_field_seek.....	1051, [3.00]
mssql_min_error_severity ...	1044, [4.00]	mysql_field_table.....	1051, [3.00]
mssql_min_message_severity .	1045, [4.00]	mysql_field_type.....	1051, [3.00]
mssql_num_fields	1044, [3.00]	mysql_free_result.....	1052, [3.00]
mssql_num_rows	1044, [3.00]	mysql_insert_id.....	1052, [3.00]
mssql_pconnect	1044, [3.00]	mysql_list_dbs.....	1052, [3.00]
mssql_query	1044, [3.00]	mysql_list_fields.....	1052, [3.00]
mssql_result	1044, [3.00]	mysql_list_tables.....	1052, [3.00]
mssql_select_db	1044, [3.00]	mysql_num_fields.....	1052, [3.00]
mt_getrandmax	952, [3.00]	mysql_num_rows.....	1053, [3.00]
mt_rand	952, [3.00]	mysql_pconnect.....	1053, [3.00]
mt_srand	952, [3.00]	mysql_query.....	1053, [3.00]
mysql_affected_rows	1049, [3.00]	mysql_result.....	1053, [3.00]
mysql_close	1049, [3.00]	mysql_select_db.....	1053, [3.00]
mysql_connect	1049, [3.00]	mysql_tablename.....	1053, [3.00]
mysql_change_user	1049, [3.13]	N	
mysql_create_db	1049, [3.00]	natcasesort.....	923, [4.00]
mysql_data_seek	1049, [3.00]	natsort.....	924, [4.00]
mysql_db_query	1050, [3.00]	new_xml doc (<i>domxml_newxml doc</i>)	
mysql_db_name	1050, [3.00]	next.....	923, [3.00]

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

- nl2br 932, [3.00]
- node_attributes 987, [4.04]
- node_children 988, [4.04]
- node_namespace 988, [4.04]
- number_format 952, [3.00]
- O**
- ob_end_clean 1068, [4.b0]
- ob_end_flush 1068, [4.b0]
- ob_get_contents 1068, [4.b0]
- ob_get_length 1069, [4.02]
- ob_implicit_flush 1069, [4.b4]
- ob_start 1069, [4.b0]
- ocibindbyname 1057, [3.04]
- ocicancel 1061, [3.04]
- ocicolumnisnull 1059, [3.04]
- ocicolumnname 1059, [3.04]
- ocicolumnprecision 1061, [4RC1]
- ocicolumnscale 1061, [4RC1]
- ocicolumnsize 1059, [3.04]
- ocicolumntype 1059, [3.04]
- ocicolumntyperaw 1062, [4RC1]
- ocicommit 1058, [3.07]
- ocidefinebyname 1057, [3.07]
- ociexecute 1057, [3.04]
- ocifetch 1058, [3.04]
- ocifetchinto 1058, [3.04]
- ocifetchstatement 1059, [3.07]
- ocifreecursor (*ocifreestatement*)
- ocifreedesc 1060, [4RC1]
- ocifreestatement 1060, [3.05]
- ociinternaldebug 1060, [3.04]
- ociloadlob 1061, [4.b4]
- ocilogoff 1057, [3.04]
- ocilogon 1057, [3.04]
- ocinewcursor 1060, [3.07]
- ocinewdescriptor 1058, [3.07]
- ocinlogon 1057, [3.07]
- ocinumcols 1058, [3.04]
- ociparse 1060, [3.04]
- ociplogon 1057, [3.07]
- ociresult 1058, [3.04]
- ocirollback 1058, [3.07]
- ocirowcount 1058, [3.07]
- ocisavelob 1061, [4.b4]
- ocisavelobfile 1061, [4.b4]
- ociserverversion 1059, [3.04]
- ocisetprefetch 1061, [3.13]
- ocistatementtype 1060, [3.04]
- ociwritelobtofile 1061, [4.b4]
- octdec 952, [3.00]
- odbc_autocommit 1053, [3.06]
- odbc_binmode 1054, [3.06]
- odbc_close 1054, [3.06]
- odbc_close_all 1054, [3.06]
- odbc_commit 1054, [3.06]
- odbc_connect 1054, [3.06]
- odbc_cursor 1054, [3.06]
- odbc_do (*odbc_exec*)
- odbc_exec 1054, [3.06]
- odbc_execute 1054, [3.06]
- odbc_fetch_into 1055, [3.06]
- odbc_fetch_row 1055, [3.06]
- odbc_field_name 1055, [3.06]
- odbc_field_type 1055, [3.06]
- odbc_free_result 1055, [3.06]
- odbc_longreadlen 1055, [3.06]

odbc_num_fields	1055, [3.06]	pdf_curveto.....	993, [3.06]
odbc_num_rows	1056, [3.06]	pdf_end_page.....	988, [3.06]
odbc_pconnect	1056, [3.06]	pdf_endpath.....	994, [3.06]
odbc_prepare	1056, [3.06]	pdf_fill.....	994, [3.06]
odbc_result	1056, [3.06]	pdf_fill_stroke.....	994, [3.06]
odbc_result_all	1056, [3.06]	pdf_get_font (<i>pdf_get_value</i>)	
odbc_rollback	1056, [3.06]	pdf_get_fontname (<i>pdf_get_parameter</i>)	
odbc_setoption	1056, [3.06]	pdf_get_fontsize (<i>pdf_get_value</i>)	
opendir	956, [3.00]	pdf_get_info.....	988, [3.06]
openlog	1072, [3.00]	pdf_get_image_height.....	990, [3.12]
ord	932, [3.00]	pdf_get_image_width.....	990, [3.12]
P		pdf_lineto.....	993, [3.06]
pack	1070, [3.00]	pdf_moveto.....	993, [3.06]
parse_str	933, [3.00]	pdf_open.....	988, [3.06]
parse_url	917, [3.00]	pdf_open_gif (<i>pdf_open_image_file</i>)	
passthru	1072, [3.00]	pdf_open_jpeg (<i>pdf_open_image_file</i>)	
pathinfo	[4.03]	pdf_open_png (<i>pdf_open_image_file</i>)	
pclose	963, [3.00]	pdf_open_tiff (<i>pdf_open_image_file</i>)	
pdf_add_annotation	997, [3.12]	pdf_open_image_file.....	996, [4RC2]
pdf_add_bookmark (<i>pdf_add_outline</i>)		pdf_open_memory_image.....	996, [3.10]
pdf_add_outline	996, [3.06]	pdf_place_image.....	997, [3.07]
pdf_add_pdflink	997, [3.12]	pdf_rect.....	993, [3.06]
pdf_add_weblink	997, [3.12]	pdf_restore.....	991, [3.06]
pdf_arc	993, [3.06]	pdf_rotate.....	992, [3.06]
pdf_begin_page	988, [3.06]	pdf_save.....	991, [3.06]
pdf_circle	993, [3.06]	pdf_scale.....	992, [3.06]
pdf_clip	995, [3.06]	pdf_set_char_spacing.....	990, [3.06]
pdf_close	988, [3.06]	pdf_set_duration.....	996, [3.06]
pdf_close_image	996, [3.07]	pdf_set_font.....	989, [3.06]
pdf_closepath	994, [3.06]	pdf_set_horiz_scaling.....	990, [3.06]
pdf_closepath_fill_stroke ...	994, [3.06]	pdf_set_info.....	988, [4.01]
pdf_closepath_stroke	994, [3.06]	pdf_set_info_author (<i>pdf_set_info</i>)	
pdf_continue_text	991, [3.06]	pdf_set_info_creator (<i>pdf_set_info</i>)	

pdf_set_info_keywords (<i>pdf_set_info</i>)	pg_dbname1062, [3.00]
pdf_set_info_subject (<i>pdf_set_info</i>)	pg_errormessage1062, [3.00]
pdf_set_info_title (<i>pdf_set_info</i>)	pg_exec1062, [3.00]
pdf_set_leading 989, [3.06]	pg_fetch_array1063, [3.00]
pdf_set_parameter 989, [4RC1]	pg_fetch_object1063, [3.00]
pdf_set_value 989, [4RC1]	pg_fetch_row1063, [3.00]
pdf_set_text_pos 990, [3.06]	pg_fieldisnull1063, [3.00]
pdf_set_text_rendering 989, [3.06]	pg_fieldname1063, [3.00]
pdf_set_text_rise 990, [3.06]	pg_fieldnum1063, [3.00]
pdf_set_transition 996, [3.06]	pg_fieldprtlen1063, [3.00]
pdf_set_word_spacing 991, [3.06]	pg_fieldsize1063, [3.00]
pdf_setdash 993, [3.06]	pg_fieldtype1064, [3.00]
pdf_setflat 992, [3.06]	pg_freeresult1064, [3.00]
pdf_setgray 995, [3.06]	pg_getlastoid1064, [3.00]
pdf_setgray_fill 995, [3.06]	pg_host1064, [3.00]
pdf_setgray_stroke 995, [3.06]	pg_loclose1064, [3.00]
pdf_setlinecap 992, [3.06]	pg_locreate1064, [3.00]
pdf_setlinejoin 992, [3.06]	pg_loopen1064, [3.00]
pdf_setlinewidth 992, [3.06]	pg_loread1064, [3.00]
pdf_setmiterlimit 992, [3.06]	pg_loreadall1065, [3.00]
pdf_setrgbcolor 996, [3.06]	pg_lounlink1065, [3.00]
pdf_setrgbcolor_fill 995, [3.06]	pg_lowrite1065, [3.00]
pdf_setrgbcolor_stroke 995, [3.06]	pg_numfields1065, [3.00]
pdf_show 989, [3.06]	pg_numrows1065, [3.00]
pdf_show_boxed 989, [4RC1]	pg_options1065, [3.00]
pdf_show_xy 989, [3.06]	pg_pconnect1065, [3.00]
pdf_stringwidth 991, [3.06]	pg_port1065, [3.00]
pdf_stroke 994, [3.06]	pg_result1066, [3.00]
pdf_translate 991, [3.06]	pg_tty1066, [3.00]
pfsckopen 970, [3.07]	phpcredits1076, [3.00]
pg_close1062, [3.00]	phpinfo1075, [3.00]
pg_cmdtuples1062, [3.00]	phpversion1075, [3.00]
pg_connect1062, [3.00]	php_logo_guid1076, [4.b4]

php_sapi_name	1076, [4.00]	putenv	1076, [3.00]
php_uname	1076, [4.02]	Q	
pi	952, [3.00]	quoted_printable_decode	933, [3.06]
popen	963, [3.00]	quotemeta	933, [3.00]
pos	924, [3.00]	R	
pow	953, [3.00]	rad2deg	953, [3.04]
preg_grep	945, [3.09]	rand	953, [3.00]
preg_match	944, [3.09]	range	924, [3.08]
preg_match_all	944, [3.09]	rawurldecode	933, [3.00]
preg_quote	945, [3.09]	rawurlencode	933, [3.00]
preg_replace	945, [3.09]	read (<i>readdir</i>)	
preg_split	945, [3.09]	readdir	956, [3.00]
prev	924, [3.00]	readfile	963, [3.00]
print	933, [3.00]	readgzfile	, [3.00]
printf	933, [3.00]	readlink	963, [3.00]
print_r	1077, [4.b1]	realpath	964, [4.b4]
pspell_add_to_personal	940, [4.02]	register_shutdown_function ..	1073, [3.04]
pspell_add_to_session	940, [4.02]	register_tick_function	1078, [4.03]
pspell_check	940, [4.02]	rename	963, [3.00]
pspell_clear_session	940, [4.02]	require	909, [3.00]
pspell_config_create	940, [4.02]	require_once	910, [4.b1]
pspell_config_ignore	940, [4.02]	reset	924, [3.00]
pspell_config_mode	941, [4.02]	restore_error_handler	1079, [4.01]
pspell_config_personal	941, [4.02]	return	910, [3.00]
pspell_config_repl	941, [4.02]	rewind	963, [3.00]
pspell_config_runtogether ...	941, [4.02]	rewinddir	956, [3.00]
pspell_config_save_repl	941, [4.02]	rmdir	963, [3.00]
pspell_new	941, [4.02]	round	953, [3.00]
pspell_new_config	941, [4.02]	rsort	925, [3.00]
pspell_new_personal	942, [4.02]	rtrim (<i>chop</i>)	
pspell_save_wordlist	942, [4.02]	S	
pspell_store_replacement	942, [4.02]	serialize	1070, [3.05]
pspell_suggest	942, [4.02]		

session_cache_limiter	970, [4.03]	split	944, [3.00]
session_decode	967, [4.b1]	spliti	944, [4.00]
session_destroy	968, [4.b1]	sprintf	934, [3.00]
session_encode	968, [4.b1]	sql_regcase	944, [3.00]
session_get_cookie_params ...	967, [4.b1]	sqrt	953, [3.00]
session_id	968, [4.b1]	srand	953, [3.00]
session_is_registered	968, [4.b1]	sscanf	934, [4.00]
session_module_name	968, [4.b1]	stat	964, [3.00]
session_name	969, [4.b1]	str_repeat	938, [4.b4]
session_register	969, [4.b1]	str_replace	938, [3.00]
session_save_path	969, [4.b1]	strchr	934, [3.00]
session_set_cookie_params	968, [4.b4]	strcasecmp	935, [3.03]
session_set_save_handler	969, [4.b4]	strcmp	935, [3.00]
session_start	969, [4.b4]	strcspn	935, [3.00]
session_unregister	969, [4.b4]	strftime	946, [3.00]
session_unset	969, [4.b4]	strip_tags	935, [3.00]
set_error_handler	1079, [4.01]	stripslashes	935, [4.b4]
set_file_buffer	969, [3.08]	stripslashes	935, [3.00]
set_magic_quotes_runtime	1077, [3.00]	stristr	936, [4.02]
set_time_limit	1077, [3.00]	strlen	936, [3.00]
setcookie	918, [3.00]	strnatcasecmp	937, [4.00]
setlocale	934, [3.00]	strnatcmp	936, [4.00]
settype	921, [3.00]	strncmp	937, [4.00]
show_source (<i>highlight_file</i>)		strpos	936, [3.00]
shuffle	925, [3.08]	strrchr	936, [3.00]
similar_text	934, [3.07]	strrev	937, [3.00]
sin	953, [3.00]	strrpos	936, [3.00]
sizeof	925, [3.00]	strspn	937, [3.00]
sleep	1073, [3.00]	strstr	937, [3.00]
socket_get_blocking	971, [4.b4]	strtok	938, [3.00]
socket_set_blocking	971, [4.b4]	strtolower	938, [3.00]
sort	925, [3.00]	strtotime	946, [3.12]
soundex	934, [3.00]	strtoupper	938, [3.00]

strtr	939, [3.00]	swf_labelframe	998, [4.01]
strval	921, [3.00]	swf_lookat	1006, [4.01]
substr	939, [3.00]	swf_modifyobject	999, [4.01]
substr_count	939, [4.00]	swf_mulcolor	998, [4.01]
substr_replace	939, [4.b4]	swf_nextid	999, [4.01]
swf_actiongeturl	1000, [4.01]	swf_oncondition	1005, [4.01]
swf_actiongotoframe	999, [4.01]	swf_openfile	998, [4.01]
swf_actiongotolabel	1001, [4.01]	swf_ortho2	1005, [4.01]
swf_actionnextframe	1000, [4.01]	swf_perspective	1005, [4.01]
swf_actionplay	1000, [4.01]	swf_placeobject	999, [4.01]
swf_actionprevframe	1000, [4.01]	swf_polarview	1005, [4.01]
swf_actionsettarget	1001, [4.01]	swf_popmatrix	1006, [4.01]
swf_actionstop	1000, [4.01]	swf_posround	1006, [4.01]
swf_actiontogglequality	1000, [4.01]	swf_pushmatrix	1006, [4.01]
swf_actionwaitforframe	1000, [4.01]	swf_removeobject	999, [4.01]
swf_addbuttonrecord	1005, [4.01]	swf_rotate	1006, [4.01]
swf_addcolor	999, [4.01]	swf_scale	1006, [4.01]
swf_closefile	998, [4.01]	swf_setfont	1003, [4.01]
swf_definebitmap	1004, [4.01]	swf_setframe	998, [4.01]
swf_definefont	1003, [4.01]	swf_shapearc	1003, [4.01]
swf_defineline	1001, [4.01]	swf_shapecurveto	1002, [4.01]
swf_definerect	1001, [4.01]	swf_shapecurveto3	1002, [4.01]
swf_definetext	1004, [4.01]	swf_shapefillbitmapclip	1002, [4.01]
swf_endbutton	1005, [4.01]	swf_shapefillbitmaptile	1002, [4.01]
swf_enddoaction	999, [4.01]	swf_shapefillloff	1001, [4.01]
swf_endshape	1003, [4.01]	swf_shapefillsolid	1002, [4.01]
swf_endsymbol	1004, [4.01]	swf_shapelinesolid	1001, [4.01]
swf_fontsize	1003, [4.01]	swf_shapelineto	1002, [4.01]
swf_fontslant	1003, [4.01]	swf_shapemoveto	1002, [4.01]
swf_fonttracking	1003, [4.01]	swf_showframe	998, [4.01]
swf_getbitmapinfo	1004, [4.01]	swf_startbutton	1004, [4.01]
swf_getfontinfo	1004, [4.01]	swf_startdoaction	999, [4.01]
swf_getframe	998, [4.01]	swf_startshape	1001, [4.01]

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

swf_startsymbol 1004, [4.01]
 swf_textwidth 1004, [4.01]
 swf_translate 1006, [4.01]
 swf_viewport 1005, [4.01]
switch 909, [3.00]
 sybase_affected_rows 1066, [3.00]
 sybase_close 1066, [3.00]
 sybase_connect 1066, [3.00]
 sybase_data_seek 1066, [3.00]
 sybase_fetch_array 1066, [3.00]
 sybase_fetch_field 1067, [3.00]
 sybase_fetch_object 1067, [3.00]
 sybase_fetch_row 1067, [3.00]
 sybase_field_seek 1067, [3.00]
 sybase_free_result 1067, [3.00]
 sybase_num_fields 1067, [3.00]
 sybase_num_rows 1067, [3.00]
 sybase_pconnect 1067, [3.00]
 sybase_query 1068, [3.00]
 sybase_result 1068, [3.00]
 sybase_select_db 1068, [3.00]
 symlink 964, [3.00]
 syslog 1072, [3.00]
 system 1072, [3.00]

T

tan 953, [3.00]
 tempnam 964, [3.00]
 textdomain 943, [4.b1]
 time 947, [3.00]
 tmpfile 964, [3.13]
 touch 964, [3.00]
 trigger_error 1079, [4.01]
 trim 939, [3.00]

U

uasort 925, [3.04]
 ucfirst 939, [3.00]
 ucwords 939, [3.03]
 uksort 925, [3.04]
 umask 965, [3.00]
 uniqid 1071, [3.00]
 unixtojd 948, [4.00]
 unlink 965, [3.00]
 unpack 1070, [3.00]
 unregister_tick_function 1078, [4.03]
 user_error (*trigger_error*)
 unserialize 1070, [3.05]
 unset 922, [3.00]
 urldecode 917, [3.00]
 urlencode 917, [3.00]
 usleep 1073, [3.00]
 usort 925, [3.04]
 utf8_decode 985, [3.06]
 utf8_encode 985, [3.06]

V

var_dump 1077, [3.05]
 virtual 1071, [3.00]

W

wddx_add_vars 982, [3.07]
 wddx_deserialize 982, [3.07]
 wddx_packet_end 982, [3.07]
 wddx_packet_start 982, [3.07]
 wddx_serialize_value 982, [3.07]
 wddx_serialize_vars 982, [3.07]
 wordwrap 940, [4.02]

X

xmlDoc 985, [4.b4]
xmlDocfile 985, [4.b4]
xmltree 985, [4.b4]
xml_error_string 984, [3.06]
xml_get_current_byte_index .. 984, [3.06]
xml_get_current_column_number 984, [3.06]
xml_get_current_line_number . 984, [3.06]
xml_get_error_code 984, [3.06]
xml_parse 984, [3.06]
xml_parser_create 982, [3.06]
xml_parser_free 984, [3.06]
xml_parser_get_option 985, [3.06]
xml_parser_set_option 984, [3.06]

xml_set_character_data_handler

..... 983, [3.06]

xml_set_default_handler..... 983, [3.06]

xml_set_element_handler..... 983, [3.06]

xml_set_external_entity_ref_handler

..... 983, [3.06]

xml_set_notation_decl_handler 983, [3.06]

xml_set_processing_instruction_handler

..... 983, [3.06]

xml_set_unparsed_entity_decl_handler

..... 983, [3.06]

Z

zend_logo_guid..... 1076, [4.b4]

zend_version..... 1076, [4.b4]

V

S

I

1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E

B.3 Befehlsreferenz

Als Befehle werden die fest in PHP eingebauten Kommandos und Kontrollstrukturen bezeichnet.

Kontrollstrukturen

if...else...elseif

Mit `if` wird eine Programmverzweigung eingeleitet, `else` leitet optional den alternativen Zweig (Nein-Zweig) ein und `elseif` setzt die Struktur mit einer Unterabfrage fort.

```
if (Bedingung) # JA-Befehl;  
if (Bedingung) { # JA-Befehle; # JA-Befehle }  
if (Bedingung) {  
    # JA-Befehle;  
} else {  
    # NEIN-Befehle;  
}  
  
if (Bedingung) {  
    # JA-Befehle;  
} else {  
    # NEIN-Befehle;  
} elseif (weitere Bedingung) {  
    # JA-Bedingung im NEIN-Zweig;  
} else  
    # NEIN-Bedingung im NEIN-Zweig;  
}
```

`if` erlaubt eine alternative Syntax, die das Umschließen von HTML-Sequenzen erleichtert:

```
if (Bedingung):  
    # JA-Befehle;  
elseif (NEIN-Bedingung):  
    # JA-im-NEIN-Zweig;  
else:  
    # NEIN-Befehle;  
endif;
```

while

Mit `while` werden Schleifen erzeugt. Die Schleife wird durchlaufen, solange die Bedingung `TRUE` ist.

```
while (Bedingung) {  
    # Befehle, solange Bedingung wahr ist;  
}
```

PHP erlaubt eine alternative Syntax, die das Umschließen von HTML-Sequenzen erleichtert:

```
while (Bedingung):  
    # Befehle, solange Bedingung wahr ist;  
endwhile;
```

Ein vorzeitiges Verlassen der Schleife ist mit dem Befehl `break` möglich, der notwendigerweise von einem `if`-Befehl ausgelöst wird.

do ... while

Bei der `do...while`-Schleife wird die Bedingung am Ende geprüft. Unabhängig von dem Ergebnis wird die Schleife also mindestens einmal durchlaufen.

```
do {  
    # Befehle-solange-Bedingung-Wahr;  
} while (Bedingung);
```

for

Mit `for`-Schleifen werden abzählbare Durchläufe gesteuert. Die Anwendung ist langsamer als `do`- oder `while`-Schleifen. Sie sollten `for` deshalb nur einsetzen, wenn die gewünschte Konstruktion mit anderen Mitteln nicht darstellbar ist.

```
for ([Startanweisung]; [Bedingung]; [Prüfanweisung]) {  
    # Befehle-solange-Bedingung-wahr;  
}
```

foreach

Diese Schleifenanweisung durchläuft ein Array oder ein assoziatives Array. Bei jedem Durchlauf wird den Laufvariablen der Wert des aktuellen Elements zugewiesen.

Im ersten Fall wird der Variablen `$value` mit jedem Durchlauf jeweils ein Element des Arrays `$array` zugewiesen:

```
foreach($array as $value) {  
    # Anweisungsblock  
}
```

Im zweiten Fall wird ein assoziatives Array erwartet. Die Schlüssel werden der Variablen `$key`, die Werte `$value` zugewiesen.

```
foreach($array as $key => $value) {  
    # Anweisungsblock  
}
```

Die Operatoren `as` und `=>` sind Bestandteil dieses Befehls.

break

Mit `break` werden andere Kontrollstrukturen unmittelbar verlassen. Die Anwendung ist nur in `if`-Anweisungen sinnvoll.

```
while ($i < 25) {  
    ...  
    if ($i == $x) { break; }  
    ...  
}
```

continue

Mit diesem Befehl wird die Programmausführung am Anfang der umgebenden Schleife fortgesetzt. Die Anwendung ist nur in `if`-Anweisungen sinnvoll.

```
while ($i < 25) {  
    ...  
  
    if ($i == $x) { continue; }  
    ...  
}
```

switch ... case

Längere Vergleiche mit `if...elseif` können unübersichtlich und fehleranfällig sein. In solchen Fällen hilft `switch`. Der Befehl prüft eine Variable auf Gleichheit mit einem Kontrollwert.

```
switch (Variable) {  
    case Kontrollwert1:  
        # Befehle-wenn-Kontrollwert1-gleich-Variable;  
        break;  
    case Kontrollwert2:  
        # Befehle-wenn-Kontrollwert2-gleich-Variable;  
        break;  
    default:  
        # Befehle-wenn-kein-Kontrollwert-gleich-Variable;  
}
```

Jeder Zweig, der angesprochen wird, sollte mit einem `break`-Befehl beendet werden.

require

Mit diesem Befehl wird der Inhalt einer Datei geladen. Die eingeschlossene Datei verhält sich so, als wäre der enthaltene Text direkt ins Skript geschrieben worden. Sie können diesen Befehl nicht innerhalb von Schleifen anwenden.

```
void require(string filename)
```

include

`include` lädt eine Datei und führt diese separat aus. Die Anwendung in Schleifen ist deshalb zulässig. Dabei wird die Datei immer wieder geladen und interpretiert.

```
void include(string filename)
```

require_once

Mit diesem Befehl wird der Inhalt einer Datei geladen. Die eingeschlossene Datei verhält sich so, als wäre der enthaltene Text direkt geschrieben worden. Mit `require_once` wird die Datei nur einmal geholt, auch wenn die Angabe mehrfach erfolgt.

```
require_once(string filename)
```

include_once

`include_once` lädt eine Datei nur einmal und führt diese separat aus. Die Anwendung in Schleifen ist deshalb zulässig. Sie müssen hinter Bedingungen (`if`, `for` usw.) Blockkennzeichen setzen, damit der gesamte Inhalt der eingeschlossenen Datei umschlossen wird.

```
void include_once(string filename)
```

function

Mit dieser Anweisung wird eine benutzerdefinierte Funktion eingeleitet.

```
function ([$variable [ = "Standardwert"]], ... weitere) {  
    # Befehle der Funktion;  
    [ return Rückgabewert;  
}
```

return

Diese Anweisung legt den Rückgabewert einer benutzerdefinierten Funktion fest.

```
function ([$variable [ = "Standardwert"]], ... weitere) {  
    # Befehle der Funktion;  
    return Ausdruck;  
}
```

class

Mit `class` wird eine Klassendefinition eingeleitet. Die Klasse umschließt Variablen und Funktionen. Variablen werden als Eigenschaften, Funktionen als Methoden der Klasse abgebildet.

```
class NamederKlasse {  
  
    var $eigenschaft;
```

```
function Methode([Parameter]) {  
    # Definition der Methode;  
}  
}
```

new

`new` erzeugt eine Instanz einer Klasse, also ein neues Objekt. Auf dieses Objekt können dann die Eigenschaften und Methoden der Klasse angewendet werden.

```
$mein_objekt = new Klassenname([Parameter])
```

extends

Um Klassen zu vererben, fehlt PHP die vollständige Funktionalität. Um einfache Vererbungen innerhalb der Definition zu ermöglichen, wird die Anweisung `extends` verwendet.

```
class TochterKlasse extends MutterKlasse {  
    # Definitionen;  
}
```

Variablen, Konstanten und Parameter

constant

Die Funktion erlaubt den indirekten Zugriff auf Konstanten.

```
mixed constant(string defined_constant)
```

define

Diese Funktion definiert eine Konstante. Konstanten können zur Laufzeit nicht mehr geändert werden.

```
void define(string name, mixed value)
```

defined

Diese Funktion untersucht, ob eine Konstante definiert wurde, und gibt dann `TRUE` zurück.

```
bool defined(string name)
```


B.4 Funktionsreferenz

Die Funktionsreferenz zeigt alle Funktionen, die im allgemeinen in Binärdistributionen und bei vielen Providern zur Verfügung stehen. Die Liste ist jedoch nicht vollständig. Einige Funktionen sind auf sehr spezielle Einsatzgebiete zugeschnitten. Zugleich wächst die Anzahl der Funktionen ständig. Aktuelle Informationen finden Sie auf der Website zum Buch.

Umgang mit Klassen

Die folgenden Funktionen unterstützen den Umgang mit Klassen.

call_user_method

Diese Funktion ruft die Methode einer benutzerdefinierten Klasse auf.

```
mixed call_user_method(string method_name, object obj  
                        [, mixed parameter [, mixed ...]])
```

call_user_method_array

Diese Funktion ruft die Methode einer benutzerdefinierten Klasse auf.

```
mixed call_user_method_array(string method_name, object obj [, array param])
```

get_class

Diese Funktion ermittelt den Namen der Klasse, in der das Objekt *obj* definiert wurde.

```
string get_class(object obj)
```

get_parent_class

Diese Funktion gibt den Namen der Elternklasse zurück, von der die Klasse abgeleitet wurde, zu der das Objekt *obj* gehört.

```
string get_parent_class(object obj)
```

get_class_methods

Diese Funktion gibt ein Array mit den Namen der Methoden zurück, die in der angegebenen Klasse definiert wurden.

```
array get_class_methods(string class_name)
```

get_class_vars

Diese Funktion gibt ein Array mit den Namen der Eigenschaften zurück, die in der angegebenen Klasse definiert wurden.

```
array get_class_vars(string class_name)
```

get_object_vars

Die Funktion gibt ein assoziatives Array zurück, das die Objekt-Eigenschaften eines spezifischen Objekts enthält.

```
array get_object_vars(object obj)
```

is_subclass_of

Die Funktion ermittelt, ob ein Objekt einer Klasse entstammt, die selbst Tochter der Klasse *superclass* ist.

```
bool is_subclass_of(object obj, string superclass)
```

class_exists

Mit `class_exists` wird überprüft, ob eine bestimmte Klasse definiert ist.

```
bool class_exists(string class_name)
```

method_exists

Die Funktion gibt TRUE zurück, wenn die angegebene Methode in dem Objekt *obj* definiert ist.

```
bool method_exists(object object, string method_name)
```

get_declared_classes

Diese Funktion gibt ein Array zurück, das die Namen aller deklarierten Klassen enthält. Ab PHP 4.0.1pl2, werden drei eingebaute Klassen am Anfang des Arrays zurückgegeben: `stdClass`, `OverloadedTestClass` und `Directory`.

```
array get_declared_classes(void)
```

com

`com` ist eine Pseudoklasse, mit der eine Instanz eines COM-Objekts gebildet werden kann. COM ist nur unter Windows verfügbar. Es gibt mehrere Funktionen, die verwendet werden können.

```
object comvar = new com(string component)
```

com_load

Lädt ein Modul vom angegebenen COM-Server.

```
string com_load(string module_name [, string server_name])
```

com_invoke

Führt eine Methode des angegebenen Objekts aus.

```
mixed com_invoke(resource object, string function_name  
[, mixed function_parameters, ...])
```

com_propget

Liest eine Eigenschaft eines COM-Objekts. `com_get` ist ein Alias für `com_propget`.

```
mixed com_propget(resource object, string property)
```

com_propput

Setzt eine Eigenschaft eines COM-Objekts. `com_propset` und `com_set` sind Aliase für `com_propput`.

```
mixed com_propput(resource object, string property)
```

Funktionsbehandlung

func_get_arg

Die Funktion gibt den Wert des angegebenen Parameters zurück. Die Parameter zählen nullbasiert. Diese Funktion kann nur innerhalb des Körpers einer benutzerdefinierten Funktion aufgerufen werden.

```
mixed func_get_arg(int argument_index)
```

func_get_args

Die Funktion gibt ein Array mit den Parametern zurück, die übergeben wurden. Diese Funktion kann nur innerhalb des Körpers einer benutzerdefinierten Funktion aufgerufen werden.

```
array func_get_args(void)
```

func_num_args

Gibt die Anzahl der Parameter zurück, die an eine benutzerdefinierte Funktion übergeben wurden, für die keine Parameter spezifiziert sind. Diese Funktion kann nur innerhalb

des Körpers einer benutzerdefinierten Funktion aufgerufen werden. Ein Beispiel finden Sie bei `func_get_args`.

```
int func_num_args(void)
```

call_user_func

Diese Funktion ruft eine benutzerdefinierte Funktion auf, die Parameter werden einzeln übergeben. *function_name* ist der Name einer Funktion als Zeichenkette. Die Parameter werden wie üblich angegeben.

```
mixed call_user_func(string function_name [, mixed parameter [, mixed ...]])
```

call_user_func_array

Diese Funktion ruft eine benutzerdefinierte Funktion auf, die Parameter werden als Array übergeben. *function_name* ist der Name einer Funktion als Zeichenkette. Die Parameter werden in einem Array übergeben.

```
array call_user_func_array(string function_name, array param)
```

create_function

Mit dieser Funktion wird eine anonyme Funktion auf der Grundlage der übergebenen Parameter erzeugt. Damit lassen sich in Abhängigkeit von bestimmten Bedingungen zur Laufzeit Funktionen erzeugen. Der Funktionscode sollte in einfachen Anführungszeichen übergeben werden, damit der Parser den Inhalt nicht schon beim Erzeugen der Funktion auswertet. Der Einsatz eignet sich für Funktionen, die als variable Parameter anderer Funktionen dienen, die auf mehrere Elemente angewendet werden, beispielsweise `array_walk`.

```
string create_function(string arguments, string functioncode)
```

global

Mit diesem Schlüsselwort werden globale Variablen innerhalb einer Funktion sichtbar gemacht, was normalerweise nicht der Fall ist. Es kann eine Liste von Variablen, getrennt durch Kommata, angegeben werden.

```
void global varname [, varname]
```

static

Variablen innerhalb einer Funktion sind immer lokal. Sie können den Inhalt einer lokalen Variablen erhalten, indem das Schlüsselwort `static` benutzt wird. Der Wert bleibt nur zur Laufzeit eines Skripts erhalten. Am Ende des Skripts gehen alle Werte verloren.

```
void static varname [, varname]
```

\$

In PHP gibt es dynamische Variablen, die keinen Wert, sondern den Namen einer anderen Variablen enthalten. Die Schreibweise beim indirekten Zugriff:

```
$$name
```

Typkonvertierungen

(double)

Mit (double) wird der Inhalt einer Variablen *varname* im Fließkommaformat dargestellt. Alternativ kann auch (real) geschrieben werden.

```
(double) varname
```

(integer)

Mit (integer) wird der Inhalt einer Variablen *varname* im Integer-Format dargestellt. Alternativ kann auch (int) geschrieben werden.

```
(integer) varname
```

(string)

Mit (string) wird der Inhalt einer Variablen *varname* als Zeichenkette dargestellt.

```
(string) varname
```

(array)

Mit (array) wird der Inhalt einer Variablen *varname* als Array dargestellt.

```
(array) varname
```

(object)

Mit (object) wird der Inhalt einer Variablen *varname* als Objekt dargestellt. Das resultierende Objekt enthält eine Eigenschaft mit dem Namen *scalar*, die den Inhalt der Variablen enthält.

```
(object) varname
```

(boolean)

Mit (boolean) wird der Inhalt einer Variablen *varname* als Boolescher Wert dargestellt (TRUE oder FALSE, intern 1 oder 0). Alternativ kann (bool) geschrieben werden.

```
(boolean) varname
```

URL und HTML

parse_url

`parse_url` zerlegt ein URL und gibt die Komponenten "scheme", "host", "port", "user", "pass", "path", "query" und "fragment" zurück. Die Bestandteile werden Indizes eines assoziativen Arrays.

```
array parse_url(string url)
```

urldecode

`urldecode` dekodiert eine Zeichenkette, die URL-kodiert war. Die URL-Kodierung nutzt `%##`-Zeichen zur Darstellung von nicht alphanumerischen Zeichen. `##` entspricht dem Hexadezimalcode des Zeichens im erweiterten Zeichensatz. `+` wird durch Leerzeichen ersetzt.

```
string urldecode(string urlstring)
```

urlencode

`urlencode` kodiert eine Zeichenkette so, dass sie per URL übertragen werden kann. Dabei werden alle nicht alphanumerischen Zeichen außer dem Unterstrich durch die Zeichenkombination `%##` ersetzt, wobei `##` für den Hexadezimalcode steht. Das Leerzeichen wird durch `+` ersetzt. Wenn Sie stattdessen `%20` für das Leerzeichen benötigen, nutzen Sie `rawurlencode`.

```
string urlencode(string urlstring)
```

base64_encode

`base64_encode` kodiert Daten nach dem MIME Base64-Verfahren. Mehr Informationen dazu finden Sie in der RFC 2045 Sektion 6.8.

```
string base64_encode(string data)
```

base64_decode

`base64_decode` dekodiert Daten aus dem Base64-Format. Die Funktion dekodiert Daten aus dem Base64-Format, das vor allem bei der Übertragung von Binärdateien per E-Mail zum Einsatz kommt. Siehe auch `base64_encode`, RFC 2045 Sektion 6.8.

```
string base64_decode(string encoded_data)
```

getallheaders

Die Funktion ermittelt alle HTTP-Header, die Bestandteil der aktuellen Anforderung waren. Die Funktion steht nur zur Verfügung, wenn PHP als Apache-Modul läuft, nicht jedoch in der CGI-Version.

```
array getallheaders(void)
```

header

header sendet eine Zeile eines HTTP-Headers. HTTP ist in RFC 2068 spezifiziert. Header müssen vor jeder anderen Ausgabe im Skript erstellt und gesendet werden. Wenn Sie eine Fehlermeldung erhalten, dass Header bereits gesendet wurden, suchen Sie nach Leerzeichen usw., die vielleicht vor dem Skript-Tag stehen.

```
int header(string headerstring)
```

headers_sent

Die Funktion gibt TRUE zurück, wenn die Header bereits gesendet wurden.

```
bool headers_sent(void)
```

setcookie

setcookie sendet ein Cookie an den Browser. Cookies werden im Header übertragen. Der Einsatz ist nur möglich, wenn noch keine Daten an den Browser übertragen wurden.

```
int setcookie(string name, string value, int expire  
               [, string path] [, string domain]  
               [, string secure])
```

Variablenfunktionen

gettype

gettype ermittelt den Typ einer Variablen *var*. Bei nicht definierten Variablen gibt die Funktion NULL zurück. Wenn Sie Funktionen auf Existenz testen möchten, bieten sich die Funktionen `function_exists` und `method_exists` an.

```
string gettype(mixed var)
```

intval

`intval` ermittelt den Ganzzahlwert einer Variablen zur angegebenen Basis *base*. Der Standardwert ist 10, diese Angabe ist optional. Sie können die Funktion auch nutzen, um Zahlenwerte aus Zeichenketten zu extrahieren, wenn die Zeichenkette mit dem Zahlenwert beginnt. Der Rest wird ignoriert. Ist keine Zahl vorhanden, wird 0 zurückgegeben.

```
int intval(mixed var [, int base])
```

doubleval

`doubleval` gibt den Inhalt einer Variablen als doppelt genaue Gleitkommazahl wieder. Sie müssen ein Skalar verwenden, Objekte oder Arrays sind nicht zulässig.

```
double doubleval(mixed var)
```

empty

`empty` ermittelt, ob eine Variable leer ist. Die Funktion gibt TRUE zurück, wenn die Variable leer ist. Beachten Sie, dass eine Zeichenkettenvariable als »leer« gilt, wenn sie eine leere Zeichenkette enthält, aber dennoch definiert ist. Eine numerische Variable ist »leer«, wenn sie die Zahl 0 enthält. Um zu prüfen, ob eine Variable definiert wurde, verwenden Sie `isset`.

```
bool empty(mixed var)
```

is_array

Die Funktion `is_array` gibt TRUE zurück, wenn die zu überprüfende Variable ein Array ist.

```
bool is_array(mixed var)
```

is_bool

Die Funktion ermittelt, ob eine Variable einen Booleschen Wert enthält, also eine der Konstanten TRUE oder FALSE.

```
bool is_bool(mixed var)
```

is_double

Die Funktion `is_double` gibt TRUE zurück, wenn der Zahlenwert, den die Variable speichert, doppelte Genauigkeit hat.

```
bool is_double(mixed var)
```

is_float

Die Funktion `is_float` gibt `TRUE` zurück, wenn die zu überprüfende Variable eine Zahl doppelter Genauigkeit ist. Die Funktion entspricht exakt `is_double`.

```
bool is_float(mixed var)
```

is_int

`is_int` ermittelt, ob eine Ganzzahl (Integer) vorliegt. Die Funktion ist ein Alias für `is_long`.

```
bool is_int(mixed var)
```

is_integer

`is_integer` ermittelt, ob eine Ganzzahl (Integer) vorliegt. Die Funktion ist ein Alias für `is_long`. Wenn mit Gleitkommazahlen operiert wird, entstehen daraus durch automatische Typumwandlungen keine Integerwerte. Setzen Sie dazu die Funktion `intval` ein.

```
bool is_integer(mixed var)
```

is_long

`is_long` ermittelt, ob eine Ganzzahl (Integer) vorliegt.

```
bool is_long(mixed var)
```

is_null

Diese Funktion gibt `TRUE` zurück, wenn die getestete Variable oder der getestete Ausdruck `NULL` ist, andernfalls `FALSE`. Die Aussage `NULL` (leer) ist ein definierter Zustand; undefinierte Variablen sind nicht `NULL`.

```
bool is_null(mixed var)
```

is_numeric

Diese Funktion gibt `TRUE` zurück, wenn der Wert der Variablen `var` numerisch ist.

```
bool is_numeric(mixed var)
```

is_object

`is_object` gibt `TRUE` zurück, wenn die Variable ein Objekt ist.

```
bool is_object(mixed var)
```

is_real

`is_real` gibt TRUE zurück, wenn die Variable eine Zahl doppelter Genauigkeit ist. Die Funktion ist ein Alias für `is_double`.

```
bool is_real(mixed var)
```

is_resource

Diese Funktion gibt TRUE zurück, wenn die Variable eine Handle ist.

```
bool is_resource(mixed var)
```

is_scalar

Diese Funktion gibt TRUE zurück, wenn die Variable kein Array ist (dies wird als »scalar« bezeichnet).

```
bool is_scalar(mixed var)
```

is_string

Die Funktion `is_string` gibt TRUE zurück, wenn es sich um eine Zeichenkette (String) handelt.

```
bool is_string(mixed var)
```

isset

Mit `isset` prüfen Sie, ob eine Variable bereits definiert wurde. Die Variable kann trotzdem leer sein.

```
bool isset(mixed var)
```

settype

`settype` setzt den Typ einer Variablen und wandelt den Inhalt entsprechend um. Wenn die Umwandlung erfolgreich war, wird TRUE zurückgegeben, andernfalls FALSE.

```
int settype(string var, string type)
```

strval

`strval` wandelt die Variable `var` in eine Zeichenkette um. Sie können nur Skalare angeben, Arrays und Objekte sind nicht zulässig.

```
string strval(mixed var)
```

unset

unset setzt eine Variable auf den undefinierten Zustand zurück. Die Funktion gibt immer TRUE zurück.

```
int unset(mixed var)
```

Array-Funktionen

array

array erzeugt aus gegebenen Werten ein Array. Für assoziative Arrays kann der =>-Operator verwendet werden. Dies ist eigentlich ein Sprachkonstrukt.

```
array array(mixed value [, mixed key => value] [, ...])
```

array_walk

array_walk wendet eine benutzerdefinierte Funktion auf jedes Element eines Arrays an.

```
int array_walk(array arr, string func)
```

arsort

arsort sortiert ein Array rückwärts. Die Zuordnung der Indizes zu den Elementen bei assoziativen Arrays bleibt erhalten. Die Sortierung erfolgt nach Elementen, nicht nach Schlüsseln.

```
void arsort(array arraytosort)
```

asort

asort sortiert ein Array vorwärts (aufsteigend). Die Zuordnung der Indizes bleibt dabei erhalten.

```
void asort(array arraytosort)
```

count

count ermittelt die Anzahl der Elemente eines Array. Es kann prinzipiell auch jeder andere Variablentyp angegeben werden. Die Funktion gibt 1 zurück, wenn *arrayvar* kein Array ist. Die Funktion gibt 0 zurück, wenn die Variable nicht definiert war.

```
int count(mixed arrayvar)
```

current

`current` gibt das aktuelle Element eines Arrays zurück. Der interne Zeiger wird dabei nicht bewegt, das Array wird nicht verändert.

```
mixed current(array arrayvar)
```

each

`each` gibt das nächste Schlüssel/Werte-Paar eines assoziativen Arrays zurück. Zurückgegeben werden vier Elemente mit den Indizes 0, 1, `key` und `value`.

```
array each(array arrayvar)
```

end

`end` setzt den internen Zeiger eines Arrays auf das letzte Element.

```
mixed end(array arrayvar)
```

key

`key` ermittelt einen Schlüssel von der aktuellen Position des internen Zeigers eines Arrays.

```
mixed key(array arrayvar)
```

ksort

`ksort` sortiert ein Array aufsteigend anhand der Schlüssel. Die Zuordnung zu den Elementen bleibt dabei erhalten. Die Anwendung ist nur für assoziative Arrays sinnvoll.

```
int ksort(array arraytosort)
```

krsort

Sortiert ein Array wie `ksort`, aber in absteigender Ordnung.

```
int krsort(array arraytosort)
```

list

`list` weist einer Gruppe Variablen mehrere Werte zu, meist zur Auflösung eines Arrays oder ähnlicher Auflistung.

```
void list(mixed var [, mixed var [, ...]])
```

natcasesort

Diese Funktion sortiert ein Array in natürlicher Reihenfolge der Elemente. Groß- und Kleinschreibung wird nicht berücksichtigt. Das übergebene Array wird direkt geändert.

Als natürlich gilt hier die Sortiermethode, wie sie ein Mensch empfinden würde (2 vor 12) , nicht die übliche zeichenweise Vorgehensweise (12 vor 2).

```
void natcasesort(array arrayvar)
```

natsort

Diese Funktion sortiert ein Array in natürlicher Reihenfolge der Elemente. Groß- und Kleinschreibung wird nicht berücksichtigt. Das übergebene Array wird direkt geändert.

```
void natsort(array arrayvar)
```

next

`next` setzt den internen Zeiger eines Elements um eine Position weiter und gibt das Element an der neuen Position zurück. Wenn das Array keine Elemente mehr enthält, wird `FALSE` zurückgegeben.

```
mixed next(array arrayvar)
```

pos

`pos` holt das aktuelle Element eines Arrays. Dies ist ein Alias für `current`.

```
mixed pos(array arrayvar)
```

prev

`prev` setzt den internen Zeiger eines Arrays um ein Element zurück. Wenn sich der Zeiger über den Anfang des Arrays bewegt, wird `FALSE` zurückgegeben. Die Funktion gibt außerdem das Element zurück, auf das der Zeiger nach der Operation zeigt.

```
mixed prev(array arrayvar)
```

range

Die Funktion `range` erzeugt ein Array mit Ganzzahlen, die aus dem angegebenen Zahlenbereich stammen, die Werte *low* und *high* inklusive.

```
array range(int low, int high)
```

reset

`reset` setzt den internen Zeiger eines Arrays auf das erste Element. Das erste Element des Arrays wird zurückgegeben.

```
mixed reset(array arrayvar)
```

rsort

`rsort` sortiert ein Array absteigend (rückwärts). Die Funktion geht davon aus, dass es sich um ein eindimensionales Array handelt.

```
void rsort(array arrayvar)
```

shuffle

`shuffle` bringt ein Array in »Unordnung«. Die Funktion ordnet die Elemente mit Hilfe des Zufallsgenerators neu.

```
void shuffle(array arraytoshuffle)
```

sizeof

`sizeof` ermittelt die Anzahl der Elemente eines Arrays. Diese Funktion ist ein Alias zu `count`.

```
int sizeof(array arrayvar)
```

sort

`sort` sortiert ein eindimensionales Array aufsteigend.

```
void sort(array arrayvar)
```

uasort

`uasort` sortiert ein Array anhand einer selbst definierten Funktion `cmp_function`. Die Funktion muss einen Vergleich zwischen zwei Elementen ausführen und -1, 0 oder 1 für kleiner, gleich oder größer zurückgeben.

```
void uasort(array arrayvar, function cmp_function)
```

uksort

`uksort` sortiert ein assoziatives Array anhand der Schlüssel und mit Hilfe einer selbst definierten Funktion `cmp_function`. Die Funktion muss einen Vergleich zwischen zwei Elementen ausführen und -1, 0 oder 1 für kleiner, gleich oder größer zurückgeben.

```
void uksort(array arrayvar, function cmp_function)
```

usort

`usort` sortiert ein eindimensionales Array anhand der Werte und mit Hilfe einer selbst definierten Funktion `cmp_function`. Die Funktion muss einen Vergleich zwischen zwei Elementen ausführen und -1, 0 oder 1 für kleiner, gleich oder größer zurückgeben.

```
void usort(array arrayvar, function cmp_function)
```

array_count_values

`array_count_values` erstellt ein Array, das die Elemente des übergebenen Arrays als Schlüssel und die Häufigkeit des Auftretens als Wert enthält.

```
array array_count_values(array input)
```

array_diff

Die Funktion `array_diff` vergleicht zwei oder mehrere Arrays miteinander und gibt ein Array zurück, das die Unterschiede (Differenzmenge) enthält. Das Resultat enthält für die übereinstimmenden Stellen jedoch immer noch Leerstellen, die sich je nach Ausgabe-funktion verschieden auswirken.

```
array array_diff(array arr1, array arr2 [, array arrX, ...])
```

array_flip

Die Funktion vertauscht alle Schlüssel mit den Werten in einem assoziativen Array. Sie kann keine mehrfach verschachtelten Arrays vertauschen, sondern nur skalare Elemente.

```
array array_flip(array input)
```

array_intersect

Diese Funktion liefert die Durchschnittsmenge aller als Parameter übergebenen Arrays.

```
array array_intersect(array arr, array arr2 [, array arr3, ...])
```

array_keys

`array_keys` gibt die Schlüssel eines zweidimensionalen Arrays als eindimensionales Array zurück. Der Parameter *search_value* ist optional. Wird er angegeben, werden nur jene Schlüssel zurückgegeben, denen exakt der entsprechende Wert zugeordnet ist.

```
array array_keys(array input [, mixed search_value])
```

array_merge

`array_merge` verbindet zwei Arrays, indem ein Array an das andere angehängt wird. Wenn die Funktion auf zweidimensionale assoziative Arrays angewendet wird und gleiche Schlüssel auftreten, überschreibt der zuletzt gelieferte Wert den vorhergehenden.

```
array array_merge(array array1, array array2 [, array arrayX [, ...]])
```

array_merge_recursive

Diese Funktion verbindet zwei oder mehrere Arrays. Dabei wird ein Array immer an das vorhergehende gehängt. Wenn die Eingabearrays dieselben Schlüssel haben, werden

diese Schlüssel zusammengefasst und aus den zugehörigen Werten wird ein Array gebildet. Dieser Prozess läuft rekursiv ab.

```
array array_merge_recursive(array arr1, array arr2 [, array arrX])
```

array_multisort

Diese Funktion sortiert ein mehrdimensionales oder mehrere eindimensionale Arrays. Das Array wird im Original geändert, die Funktion gibt TRUE zurück, wenn die Sortierung erfolgreich war.

```
bool array_multisort(array input [, mixed arg [, mixed ...]])
```

array_pad

array_pad vergrößert ein Array am Anfang oder am Ende. Dabei werden die neuen Elemente mit *value* aufgefüllt.

```
array array_pad(array arrayval, int size, mixed value)
```

array_pop

array_pop entfernt das letzte Element eines Arrays und gibt es zurück. Das Array ist danach ein Element kleiner. Die Funktion arbeitet nur mit eindimensionalen Arrays. Bei assoziativen Arrays werden die Schlüssel ignoriert und gehen verloren.

```
mixed array_pop(array array)
```

array_push

array_push fügt ein Element am Ende eines Arrays hinzu. Das Array wächst dabei um ein Element.

```
int array_push(array arrvar, mixed var [, mixed var2][,...])
```

array_reverse

array_reverse dreht ein Array um. Das letzte Element steht nun an der ersten Position und umgekehrt.

```
array array_reverse(array arrayvar)
```

array_shift

array_shift entfernt ein Element vom Anfang und gibt es zurück. Alle Elemente rutschen eine Position nach unten. Das Array ist nun um ein Element kleiner. Die Funktion arbeitet nur mit eindimensionalen Arrays. Bei assoziativen Arrays werden die Schlüssel ignoriert und gehen verloren.

```
mixed array_shift(array arrayvar)
```

array_rand

Diese Funktion liefert ein oder mehrere Elemente eines eindimensionalen Arrays zufalls-gesteuert zurück.

```
array array_rand(array arrayvar)
```

array_unique

Die Funktion untersucht ein Array und entfernt alle doppelten Elemente.

```
array array_unique(array arrayvar)
```

array_unshift

`array_unshift` fügt mehrere Elemente am Anfang eines Arrays ein. Die Elemente werden einzeln, nicht als Array, übergeben. Das resultierende Array wächst dabei um mehrere Elemente.

```
int array_unshift(array arrayvar, mixed val [, mixed val2,]...)
```

array_slice

`array_slice` gibt einige (bestimmte) Elemente des Arrays zurück. Die Elemente werden anhand der Position im Array ausgewählt.

```
array array_slice(array input, int offset [, int length])
```

array_splice

`array_splice` ersetzt einige (bestimmte) Elemente des Arrays. Die durch die Position im Array beschriebenen Elemente werden durch bestimmte Werte ersetzt.

```
array array_splice(array arrayvar, int offset [, int length] [, array replacement])
```

array_sum

Die Funktion summiert die Werte eines assoziativen Arrays oder die Elemente eines einfachen Arrays.

```
double array_sum(array tosum)
```

array_values

`array_values` gibt alle Werte eines zweidimensionalen, assoziativen Arrays zurück.

```
array array_values(array arrayvar)
```

compact

`compact` überführt Variablen und deren Werte in ein Array. Die Funktion kann zum Prüfen von Variablen genutzt werden, außerdem vereinfacht sie die Parameterübergabe. Alternativ kann auch als Parameter ein Array angegeben werden, dessen Elemente die Namen von Variablen bezeichnen.

```
array compact(string varname | array arrnames [, ...])
```

extract

Diese Funktion erzeugt Variablen aus einem Array, indem die Namen in der Symboltabelle angelegt werden.

```
void extract(array arrayvar [, int extract_type [, string prefix]])
```

in_array

`in_array` untersucht ein Array auf das Vorhandensein von Elementen mit bestimmten Eigenschaften und gibt einen Booleschen Wert zurück. Die Funktion arbeitet nur mit eindimensionalen Arrays oder ignoriert bei assoziativen Arrays den Schlüssel.

```
bool in_array(mixed needle, array haystack)
```

Zeichenkettenfunktionen

addslashes

`addslashes` versieht in einer Zeichenkette bestimmte Elemente mit einem Backslash. Die Zeichenkette wird modifiziert zurückgegeben.

```
string addslashes(string str)
```

addcslashes

`addcslashes` versieht in einer Zeichenkette bestimmte Elemente mit einem Backslash. Die Zeichenkette wird modifiziert zurückgegeben.

```
string addcslashes(string str, string charlist)
```

bin2hex

`bin2hex` konvertiert eine Zeichenkette in eine Folge von Hexadezimalzahlen, die den Hexcodes der einzelnen Zeichen entsprechen.

```
string bin2hex(string str)
```

chop

chop entfernt alle Whitespaces vom Ende der Zeichenkette und gibt die modifizierte Zeichenkette zurück.

```
string chop(string str)
```

chr

chr gibt das Zeichen zu einem gegebenen ASCII-Wert zurück.

```
string chr(int ascii)
```

chunk_split

Die Funktion chunk_split teilt eine Zeichenkette in kleinere Stücke konstanter Länge.

```
string chunk_split(string string [, int chunklength] [, string endchar])
```

convert_cyr_string

convert_cyr_string konvertiert Zeichen zwischen kyrillischen Zeichensätzen.

```
string convert_cyr_string(string str, string from, string to)
```

count_chars

Ermittelt die Anzahl gleicher Zeichen in einer Zeichenkette.

```
mixed count_chars(string string [,int mode])
```

crc32

Berechnet das CRC32-Polynom (Prüfsumme) einer Zeichenkette.

```
int crc32(string str)
```

crypt

crypt verschlüsselt eine Zeichenkette *str* nach DES (Standard UNIX DES).

```
string crypt(string str [, string salt])
```

echo

echo gibt Zeichenketten aus. Es können beliebig viele Parameter angegeben werden, die alle nacheinander durch Komma getrennt werden. Innerhalb von Zeichenketten mit doppelten Anführungszeichen werden Variablen erkannt und ausgewertet, bei einfachen Anführungszeichen erfolgt dies nicht.

```
void echo(string arg1 [, string argn, ...])
```

explode

`explode` zerteilt eine Zeichenkette *str* anhand eines Separators *separator* und gibt die Elemente als Array zurück.

```
array explode(string separator, string str)
```

get_html_translation_table

Diese Funktion gibt die Übersetzungstabelle für die HTML-Entitäten zurück. In dieser Tabelle steht die direkte Zuordnung zwischen Zeichencodes und HTML-Darstellung.

```
array get_html_translation_table(int table)
```

get_meta_tags

`get_meta_tags` liest eine Datei, ermittelt darin alle <META>-Tags und gibt die gefundenen Tags als assoziatives Array zurück. Der Name wird zum Schlüssel, der Inhalt zum Wert des Arrays.

```
array get_meta_tags(string filename [, int use_include_path])
```

htmlspecialchars

`htmlspecialchars` wandelt Sonderzeichen in die entsprechenden HTML-Codes um.

```
string htmlspecialchars(string strvar [, int style])
```

htmlentities

`htmlentities` wandelt alle bekannten Sonderzeichen in die entsprechenden HTML-Codes um. Der ISO-8859-1-Zeichensatz wird zur Erkennung der Sonderzeichen verwendet.

```
string htmlentities(string htmlstr)
```

implode

`implode` verbindet die Elemente eines Arrays mit einem Separator zu einer Zeichenkette.

```
string implode(string separator, array arrayvar)
```

join

`join` ist ein Alias für `implode`.

```
string join(string separator, array arrayvar)
```

levenshtein

Diese Funktion realisiert eine Ähnlichkeitsanalyse nach dem Levenshtein-Algorithmus. Dabei wird nach einem bestimmten Verfahren die Anzahl der Schritte bestimmt, die zur Umwandlung einer Zeichenkette in eine andere benötigt werden. Je geringer diese Zahl ist, umso ähnlicher sind sich die Zeichenketten.

```
string levenshtein(string string1, string string2)  
string levenshtein(string string1, string string2  
                   [, string cost_insert] [, string cost_replace]  
                   [, string cost_delete])
```

ltrim

`ltrim` entfernt Whitespaces vom Anfang einer Zeichenkette.

```
string ltrim(string stringvar)
```

md5

`md5` errechnet den MD5-Code einer Zeichenkette.

```
string md5(string stringvar)
```

metaphone

Die Funktion erzeugt einen »Klangschlüssel« eines Wortes. Damit können phonetische Vergleiche durchgeführt werden.

```
string metaphone(string stringvar)
```

n12br

`n12br` fügt vor jedem Zeilenumbruch (Newline, `\n`) den HTML-Code `
` ein. Das `\n`-Zeichen bleibt dabei erhalten.

```
string n12br(string stringvar)
```

ord

`ord` gibt den ASCII-Wert eines Zeichens zurück. Wird eine Zeichenkette mit mehr als einem Zeichen übergeben, wird der Wert nur für das erste Zeichen ermittelt.

```
int ord(string stringvar)
```

parse_str

`parse_str` durchsucht eine Zeichenkette, als handelte es sich um den GET-Parameter (so genannter »QueryString«), und gibt die Werte als Elemente eines Arrays zurück.

```
void parse_str(string stringvar)
```

print

`print` gibt eine Zeichenkette aus.

```
int print(string argument)
```

printf

`printf` gibt eine Zeichenkette aus und beachtet dabei Formatieranweisungen.

```
int printf(string format, mixed arguments [, ...])
```

quoted_printable_decode

`quoted_printable_decode` konvertiert eine Zeichenkette, die »Quoted Printable«-codiert ist, in 8-Bit-Werte. »Quoted Printable« wird bei der Codierung von MIME-Anhängen verwendet.

```
string quoted_printable_decode(string stringvar)
```

quotemeta

`quotemeta` stellt Zeichen mit besonderer Bedeutung ein Backslash voran.

```
int quotemeta(string stringvar)
```

rawurldecode

`rawurldecode` decodiert URL-codierte Zeichenketten.

```
string rawurldecode(string stringvar)
```

rawurlencode

`rawurlencode` kodiert Zeichenketten nach RFC 1738. Dabei werden alle Sonderzeichen außer »-« und »_« durch % und einen zweistelligen Hexadezimalwert des ASCII-Codes des Zeichens ersetzt.

```
string rawurlencode(string stringvar)
```

rtrim

Diese Funktion entfernt am Ende einer Zeichenkette (Alias zu `chop`).

```
string rtrim(string stringvar)
```

setlocale

`setlocale` setzt Lokalisierungsinformationen. Damit lassen sich allgemeine Angaben wie Geldwerte, Zahlen oder ein Datum in einer landestypischen Form ausgeben.

```
string setlocale(string category, string locale)
```

similar_text

`similar_text` ermittelt eine Ähnlichkeit zwischen zwei Zeichenketten. Zurückgegeben wird die Anzahl übereinstimmender Zeichen absolut oder in Prozent (mit der Angabe *percent*).

```
int similar_text(string str1, string str2 [, double percent])
```

soundex

`soundex` berechnet den Lautwert einer Zeichenkette für Ähnlichkeitsvergleiche.

```
string soundex(string stringvar)
```

sscanf

Diese Funktion untersucht eine Eingabezeichenkette auf bestimmte Formatierungsmuster und weist erkannte Werte den Variablen zu. Dieser Vorgang ist praktisch die Umkehrung zu `sprintf`.

```
mixed sscanf(string stringvar, string pattern [, var1] [...])
```

sprintf

`sprintf` gibt eine Zeichenkette formatiert zurück. Das Argument *pattern* bestimmt, wie die einzelnen Argumente *arguments* formatiert werden.

```
string sprintf(string pattern, mixed arguments [, ...])
```

Das Steuerzeichen in *pattern* hat folgenden Aufbau:

```
%[fill][-][width][.][precision]
```

strchr

`strchr` findet das erste Auftreten der Zeichen *chars* in der Zeichenkette *stringvar*.

```
string strchr(string stringvar, string chars)
```

strcasecmp

Die Funktion vergleicht zwei Zeichenketten. Groß- und Kleinschreibung wird nicht berücksichtigt.

```
int strcasecmp(string string1, string string2)
```

strncasecmp

Die Funktion vergleicht zwei Zeichenketten. Groß- und Kleinschreibung wird nicht berücksichtigt. Der Vergleich wird nur für die Anzahl *number* Zeichen durchgeführt, die angegeben wurden.

```
int strncasecmp(string string1, string string2, int number)
```

strcmp

strcmp vergleicht zwei Zeichenketten auf Binärebene. Der Vergleich berücksichtigt daher Groß- und Kleinschreibung. Wenn *str1* kleiner ist als *str2*, wird -1 zurückgegeben, bei Gleichheit dagegen 0, in allen anderen Fällen +1.

```
int strcmp(string str1, string str2)
```

strcspn

Die Funktion strcspn gibt die Anzahl der Zeichen in *str1* zurück, die *nicht* in *str2* enthalten sind.

```
int strcspn(string str1, string str2)
```

strip_tags

strip_tags entfernt alle PHP- und HTML-Tags aus einer Zeichenkette. Ein Laufzeitfehler tritt auf, wenn sinnlose oder unvollkommene Tags gefunden werden. Im Parameter *excludelist* kann angegeben werden, welche Tags nicht gefunden werden sollen. Die Angabe erfolgt als kommaseparierte Liste.

```
string strip_tags(string stringvar [, string excludelist])
```

stripcslashes

stripcslashes kehrt die Funktion addcslashes um und entfernt die Backslashes vor Sonderzeichen.

```
string stripcslashes(string stringvar)
```

stripslashes

`stripslashes` kehrt die Funktion `addslashes` um und entfernt die Backslashes vor allen Sonderzeichen.

```
string stripslashes(string stringvar)
```

stristr

`stristr` findet das erste Auftreten einer Zeichenkette *needle* in der Zeichenkette *haystack*. Zurückgegeben wird der Teil aus *haystack*, ab dem die Übereinstimmung auftritt. Wird keine Übereinstimmung gefunden, gibt die Funktion `FALSE` zurück. Wenn *needle* kein Zeichen ist, wird der Wert als ASCII-Wert interpretiert und daraus ein Zeichen ermittelt. Die Funktion berücksichtigt nicht Groß- und Kleinschreibung.

```
string stristr(string haystack, string needle)
```

strlen

`strlen` ermittelt die Länge einer Zeichenkette *stringvar*.

```
int strlen(string stringvar)
```

strrpos

`strrpos` findet das letzte Auftreten einer Zeichenkette *charneedle* in der Zeichenkette *haystack*. Beachten Sie, dass *charneedle* nur ein Zeichen sein darf, keine Zeichenkette. Wenn dennoch eine Zeichenkette übergeben wird, interpretiert die Funktion nur das erste Zeichen. Wird keine Übereinstimmung gefunden, gibt die Funktion `FALSE` zurück.

```
int strrpos(string haystack, string charneedle)
```

strpos

`strpos` findet das erste Auftreten der Zeichenkette *needle* in der Zeichenkette *haystack*. Der Vergleich beginnt an der mit *offset* bestimmten Position. Wird keine Übereinstimmung gefunden, gibt die Funktion `FALSE` zurück.

```
int strpos(string haystack, string needle [, int offset])
```

strnatcmp

Diese Funktion vergleicht Zeichenketten so, wie es die übliche (menschliche) Leseweise ergeben würde.

```
int strnatcmp(string string1, string string2)
```

strncmp

Die Funktion vergleicht zwei Zeichenketten, wobei die Anzahl der zu vergleichenden Zeichen eingeschränkt werden kann. Der Vergleich ist »binärsicher«. Wenn die durch `tocompare` angegebene Anzahl Zeichen übereinstimmt, gibt die Funktion `TRUE` zurück, sonst `FALSE`.

```
bool strncmp(string string1, string string2, int tocompare)
```

strnatcasecmp

Diese Funktion vergleicht Zeichenketten so, wie es die übliche Leseweise ergeben würde. Die Funktion berücksichtigt Groß- und Kleinschreibung nicht.

```
int strnatcmp(string string1, string string2)
```

strrchr

`strrchr` findet das letzte Vorkommen eines Zeichens *needle* in einer Zeichenkette *haystack*. Wird keine Übereinstimmung gefunden, gibt die Funktion `FALSE` zurück. Ansonsten wird der Teil der Zeichenkette ab dem Fundort zurückgegeben.

```
string strrchr(string haystack, string needle)
```

strrev

`strrev` dreht eine Zeichenkette um.

```
string strrev(string stringvar)
```

strspn

`strspn` sucht die Zeichen aus *str2* in *str1* und gibt die erste Position zurück, an der ein Zeichen auftritt, das in *str2* nicht enthalten ist. Die Reihenfolge der Zeichen in *str2* spielt keine Rolle. Die Funktion berücksichtigt Groß- und Kleinschreibung.

```
int strspn(string str1, string str2)
```

strstr

`strstr` findet das erste Auftreten einer Zeichenkette *needle* in der Zeichenkette *haystack*. Wird keine Übereinstimmung gefunden, gibt die Funktion `FALSE` zurück. Wenn *needle* kein Zeichen ist, wird der Wert als ASCII-Wert interpretiert und daraus ein Zeichen ermittelt.

```
string strstr(string haystack, string needle)
```

strtok

`strtok` zerlegt eine Zeichenkette anhand des Trennzeichens *delimiter* in mehrere Teile (Token). Die zu zerlegende Zeichenkette muss nur beim ersten Aufruf angegeben werden. Alle folgenden Aufrufe benötigen nur das Trennzeichen.

```
mixed strtok([string argument,] string delimiter)
```

strtolower

`strtolower` wandelt die Zeichen der Zeichenkette *stringvar* in Kleinbuchstaben um. Umlaute werden, wenn das Skript nicht auf einem lokalisierten Betriebssystem läuft, nicht bearbeitet.

```
string strtolower(string stringvar)
```

strtoupper

`strtoupper` gibt eine Zeichenkette in Großbuchstaben zurück. Umlaute werden, wenn das Skript nicht auf einem lokalisierten Betriebssystem läuft, nicht gewandelt.

```
string strtoupper(string stringvar)
```

str_pad

Diese Funktion erweitert eine Zeichenkette links, rechts oder auf beiden Seiten mit einem oder mehreren Zeichen. Als Füllstoff kann auch eine Zeichenkette verwendet werden. Angegeben wird die gewünschte Länge *length* der aufgefüllten Zeichenkette.

```
string str_pad(string str, int length [, string fill] [, int type])
```

str_repeat

Die Funktion wiederholt eine Anzahl Zeichen oder eine Zeichenkette. Die Anzahl, die angegeben wird, muss größer als 0 sein.

```
string str_repeat(string stringtorrepeat, int times)
```

str_replace

`str_replace` ersetzt alle Vorkommen von *needle* in *haystack* mit *str*. Die Funktion ersetzt auf der Basis eines Binärvergleichs.

```
string str_replace(string needle, string str, string haystack)
```

strtr

`strtr` tauscht Zeichen anhand einer Austauschliste aus. Die Zeichenkette *str* wird durchsucht, jedes Auftreten eines Zeichens in *from* wird durch das an gleicher Position befindliche Zeichen in *to* ersetzt.

```
string strtr(string stringvar, string from, string to)
```

substr

`substr` gibt den Teil einer Zeichenkette *stringvar* zurück, beginnend an der Position *start* und mit *length* Zeichen.

```
string substr(string stringvar, int start [, int length])
```

substr_count

Diese Funktion ermittelt die Anzahl des Auftretens einer Zeichenkette in einer anderen. Die Funktion unterscheidet Groß- und Kleinschreibung.

```
string substr_count(string stringvar, string search)
```

substr_replace

Die Funktion ersetzt einen bestimmten Bereich einer Zeichenkette mit den Ersatzzeichen *replace*. Der Bereich reicht von *from* bis zum Ende der Zeichenkette oder *len* Zeichen lang.

```
string substr_replace(string strvar, string replace, int from [, int len])
```

trim

`trim` entfernt Whitespaces vom Anfang und vom Ende einer Zeichenkette. Dies sind die Zeichen `\t`, `\r`, `\n`, `\v`, `\0` und das reguläre Leerzeichen.

```
string trim(string stringvar)
```

ucfirst

`ucfirst` wandelt das erste Zeichen einer Zeichenkette in einen Großbuchstaben um.

```
string ucfirst(string stringvar)
```

ucwords

`ucwords` wandelt das erste Zeichen jedes Wortes einer Zeichenkette in einen Großbuchstaben um.

```
string ucwords(string stringvar)
```

wordwrap

Die Funktion trennt nach einer angegebenen Anzahl Zeichen mit Hilfes des Trennzeichens *breakchar* ab. Wird der Parameter nicht angegeben, wird \n als Trennzeichen verwendet.

```
string wordwrap(string var [, int maxwidth [, string breakchar [, int cuttype]])
```

Funktionen für Rechtschreibprüfung

pspell_add_to_personal

Fügt der persönlichen Wortliste *dictionary* ein neues Wort *newword* hinzu.

```
int pspell_add_to_personal(int dictionary, string newword)
```

pspell_add_to_session

Fügt der persönlichen Wortliste *dictionary*, die in der aktuellen Sitzung verwendet wird, ein neues Wort *newword* hinzu.

```
int pspell_add_to_session(int dictionary, string newword)
```

pspell_check

Mit dieser Funktion wird ein Wort im Wörterverzeichnis nachgeschlagen.

```
bool pspell_check(int dictionary_link, string word)
```

pspell_clear_session

Mit dieser Funktion wird die aktuelle Sitzung in Bezug auf das verwendete Wörterbuch beendet,

```
int pspell_clear_session(int dictionary)
```

pspell_config_create

Die Funktion erzeugt eine Konfiguration, die später zur Eröffnung eines Wörterbuchs genutzt werden kann.

```
int pspell_config_create(string language  
                        [, string spelling [, string jargon [, string suggesttype]])
```

pspell_config_ignore

Verändert die aktuelle Konfiguration so, dass Wörter unterhalb der Länge *length* nicht beachtet werden.

```
int pspell_config_ignore(int dictionary, int length)
```

pspell_config_mode

Mit dieser Funktion wird das Vorschlagsverhalten geändert. Die Einstellungen entsprechen denen beim Öffnen des Wörterbuchs zulässigen.

```
int pspell_config_mode(int dictionary, int suggesttype)
```

pspell_config_personal

Diese Funktion legt eine Datei fest, in der das persönliche Wörterbuch gespeichert wird.

```
int pspell_config_personal(int dictionary, string filename)
```

pspell_config_repl

Mit dieser Funktion wird eine Datei festgelegt, in der Ersatzpaare gespeichert werden.

```
int pspell_config_repl(int dictionary, string filename)
```

pspell_config_runtogether

Mit dieser Konfigurationsfunktion wird festgelegt, das gültige Wörter zusammengesetzt werden dürfen.

```
int pspell_config_runtogether(int dictionary, boolean flag)
```

pspell_config_save_repl

Diese Funktion legt fest, ob Ersatzwörter zusammen mit der Hauptwörterliste gespeichert werden.

```
int pspell_config_save_repl(int dictionary, bool flag)
```

pspell_new

Diese Funktion lädt ein neues Verzeichnis.

```
int pspell_new(string language  
               [, string file [, string jargon [, string encoding [, int mode]]]])
```

pspell_new_config

Diese Funktion erzeugt ein neues Wörterbuch und nutzt dabei eine vorbereitete Konfiguration.

```
int pspell_new_config(int config)
```

Mehr Informationen finden Sie unter `pspell_config_create`.

pspell_new_personal

Diese Funktion lädt ein neues persönliches Wörterbuch.

```
int pspell_new_personal(string personal, string language  
                        [, string file [, string jargon [, string encoding  
                        [, int suggesttype]]]])
```

pspell_save_wordlist

Mit dieser Funktion wird die aktuelle Wörterliste in der persönlichen Wörterbuchdatei abgelegt.

```
int pspell_save_wordlist(int dictionary)
```

pspell_store_replacement

Diese Funktion speichert ein neues Ersatzwörterpaar in der Vorschlagsliste.

```
int pspell_store_replacement(int dictionary, string wrong, string correct)
```

pspell_suggest

Diese Funktion gibt einen Vorschlag für eine Schreibweise zurück.

```
array pspell_suggest(int dictionary_link, string word)
```

GNU Gettext Sprachunterstützung

bindtextdomain

Diese Funktion bindet eine Domain an eine bestimmte Sprachdatei.

```
string bindtextdomain(string domainname, string directory)
```

dcgettext

Diese Funktion überschreibt die aktuelle Domainzuordnung für eine einzige Abfrage. Außerdem kann die Kategorie ausgewählt werden.

```
string dcgettext(string domainname, string message, int category)
```

dgettext

Diese Funktion überschreibt die aktuelle Domainzuordnung für eine einzige Abfrage.

```
string dgettext(string domainname, string message)
```

gettext

Diese Funktion übersetzt eine Zeichenkette anhand der Übersetzungstabelle und der voreingestellten Domain. Wird keine Übersetzung gefunden, gibt die Funktion die Anfrage zurück.

```
string gettext(string message)
```

textdomain

Diese Funktion stellt die Standarddomain ein, die verwendet wird, wenn gettext aufgerufen wird. Die Domain muss zuvor mit der Übersetzungstabelle verbunden worden sein. Wenn die Funktion textdomain ohne Parameter aufgerufen wird, gibt sie die aktuellen Einstellungen zurück, ohne diese zu ändern.

```
int textdomain([string library])
```

Standardfunktionen für reguläre Ausdrücke

ereg

ereg durchsucht eine Zeichenkette *stringvar* nach einem regulären Ausdruck *muster*.

```
int ereg(string muster, string stringvar [, array result])
```

ereg_replace

ereg_replace sucht und ersetzt eine Zeichenkette nach einem regulären Ausdruck. Die Funktion durchsucht *stringvar* nach dem regulären Ausdruck *muster* und ersetzt gefundene Teile durch *ersatz*.

```
string ereg_replace(string muster, string ersatz, string stringvar)
```

eregi

eregi durchsucht eine Zeichenkette *stringvar* nach einem regulären Ausdruck *muster*.

```
int eregi(string muster, string stringvar [, array result])
```

eregi_replace

eregi_replace sucht und ersetzt eine Zeichenkette nach einem regulären Ausdruck.

```
string eregi_replace(string muster, string ersatz, string stringvar)
```

split

`split` zerlegt eine Zeichenkette und gibt die Fragmente als Array zurück. Die Splitfunktion *muster* ist ein regulärer Ausdruck. Der Parameter *limit* begrenzt die Anzahl der Elemente. Das letzte Element des Arrays enthält den gesamten Rest der Zeichenkette.

```
array split(string muster, string stringvar [, int limit])
```

spliti

`spliti` zerlegt eine Zeichenkette und gibt die Fragmente als Array zurück. Die Splitfunktion ist ein regulärer Ausdruck. Groß- und Kleinschreibung wird nicht berücksichtigt. *limit* schränkt die Anzahl der Trennungen ein.

```
array spliti(string muster, string stringvar [, int limit])
```

sql_regcase

`sql_regcase` überführt ein Suchmuster in einen regulären Ausdruck, der unabhängig von Groß- und Kleinschreibung ist.

```
string sql_regcase(string str)
```

PCRE-Funktionen für reguläre Ausdrücke

preg_match

`preg_match` durchsucht eine Zeichenkette nach einem regulären Ausdruck. Die Zeichenkette *subject* wird anhand des Suchmusters *muster* durchsucht. Die Funktion gibt TRUE zurück, wenn die Suche erfolgreich war. Wenn Gruppen gebildet wurden, werden die Ergebnisse in dem Array *result* zurückgegeben. *result[0]* enthält die gesamte Zeichenkette, *matches[1]* die erste Gruppe usw.

```
int preg_match(string muster, string subject [, array result])
```

preg_match_all

`preg_match_all` sucht global nach einem Suchmuster. Die Zeichenkette *subject* wird anhand des Suchmusters *muster* durchsucht. Die Funktion gibt TRUE zurück, wenn die Suche erfolgreich war. Wenn Gruppen gebildet wurden, werden die Ergebnisse in dem Array *\$result* zurückgegeben. *\$result[0]* enthält die gesamte Zeichenkette, *\$result[1]* die erste Gruppe usw.

```
int preg_match_all(string muster, string subject, array result [, int order])
```

preg_replace

`preg_replace` sucht nach einem regulären Ausdruck und ersetzt die Fundstelle. Die Zeichenkette *test* wird anhand des Suchmusters *muster* durchsucht und an den Fundstellen durch *ersatz* ersetzt. Die Anzahl der Ersetzungen kann mit *limit* begrenzt werden – dieser Parameter ist optional.

```
mixed preg_replace(mixed muster, mixed ersatz, mixed test [, int limit])
```

preg_split

`preg_split` zerlegt eine Zeichenkette anhand eines regulären Ausdrucks. Zurückgegeben wird ein Array, dessen Elemente jeweils einen Teil der Zeichenkette enthalten. Die maximale Anzahl der Elemente kann mit *limit* eingeschränkt werden.

```
array preg_split(string muster, string subject [, int limit [, int flags]])
```

preg_quote

`preg_quote` markiert alle Sonderzeichen, die in regulären Ausdrücken eine Bedeutung haben, sodass sie ihre Bedeutung verlieren. Dazu wird ein Backslash davor gesetzt.

```
string preg_quote(string stringvar [, string delimiter])
```

preg_grep

`preg_grep` gibt in einem Array Teile der Zeichenkette zurück, für die eine Übereinstimmung mit dem regulären Ausdruck gefunden wurde. Durchsucht wird das Array *input-array* elementweise.

```
array preg_grep(string muster, array inputarray)
```

Datums-, Kalender- und Zeitfunktionen

checkdate

`checkdate` prüft ein Datum auf Gültigkeit.

```
int checkdate(int month, int day, int year)
```

date

`date` formatiert eine Zeichenkette zur Ausgabe von Datum und Uhrzeit. Die ausgegebene Zeit kann *timestamp* entsprechen – fehlt diese Angabe, wird die aktuelle Zeit des Servers genommen. Die Ausgabe wird anhand eines Musters in *formatstring* erstellt.

```
string date(string formatstring [, int timestamp])
```

strftime

strftime formatiert eine Datumsangabe nach den Angaben der Region. Die ausgegebene Zeit kann *timestamp* entsprechen – fehlt diese Angabe, wird die aktuelle Zeit des Servers genommen.

```
string strftime(string formatstring [, int timestamp])
```

gmstrftime

gmstrftime formatiert eine Zeitangabe. Die Parameter entsprechen strftime. Die Zeit wird umgerechnet auf die GMT-Zeit (Greenwich Mean Time).

```
string gmstrftime(string formatstring, int timestamp)
```

getdate

getdate ermittelt Zeit- und Datumsinformationen.

```
array getdate(int timestamp)
```

gettimeofday

gettimeofday ermittelt die aktuelle Zeit.

```
array gettimeofday(void)
```

gmdate

gmdate formatiert eine GMT-Zeitangabe.

```
string gmdate(string formatstring, int timestamp)
```

localtime

Die Funktion gibt ein Array mit Zeitinformationen über die Zeit zurück, die der übergebene Zeitstempel anzeigt. Wurde kein Parameter übergeben, wird die aktuelle Server-Zeit eingesetzt.

```
array localtime([int timestamp [, bool is_associative]])
```

strtotime

Diese Funktion erwartet ein Datum mit ausgeschriebenen Namen in englischer Sprache. Zurückgegeben wird der Unix-Zeitstempel.

```
int strtotime(string time [, int now])
```

mktime

mktime ermittelt den Unix-Zeitstempel für ein gegebenes Datum.

```
int mktime(int hour, int minute, int second,  
           int month, int day, int year [, int is_daysave])
```

gmtime

gmtime ermittelt den Zeitstempel nach GMT. Die Funktion entspricht mktime, berücksichtigt aber die Differenz zu GMT.

```
int gmtime(int hour, int minute, int second,  
           int month, int day, int year, int [is_dst])
```

time

time gibt den aktuellen UNIX-Zeitstempel zurück.

```
int time(void)
```

microtime

microtime ermittelt den Zeitstempel der Systemzeit mit Mikrosekunden.

```
string microtime(void)
```

Kalenderfunktionen

jdtogregorian

jdtogregorian konvertiert eine julianische Tageszählung in ein gregorianisches Datum der Form »Monat/Tag/Jahr«.

```
string jdtogregorian(int julianday)
```

gregoriantojd

gregoriantojd konvertiert ein gregorianisches Datum in eine julianische Tageszählung.

```
int gregoriantojd(int month, int day, int year)
```

jdtojulian

jdtojulian konvertiert die julianische Tageszählung in ein julianisches Datum.

```
string jdtojulian(int julianday)
```

jdtonix

jdtonix konvertiert die julianische Tageszählung in den passenden Unix-Zeitstempel.

```
string jdtonix(int julianday)
```

unixtojd

unixtojd den Unix-Zeitstempel in die julianische Tageszählung.

```
int unixtojd([int timestamp])
```

juliantojd

juliantojd konvertiert ein julianisches Datum in die julianische Tageszählung.

```
int juliantojd(int month, int day, int year)
```

jdtojewish

jdtojewish konvertiert die julianische Tageszählung in den jüdischen Kalender.

```
string jdtojewish(int julianday)
```

jewishtojd

jewishtojd konvertiert ein jüdisches Datum in die julianische Tageszählung.

```
int jewishtojd(int month, int day, int year)
```

jdtofrench

jdtofrench konvertiert die julianische Tageszählung in den Kalender der französischen Republik.

```
string jdtofrench(int month, int day, int year)
```

frenchtojd

frenchtojd konvertiert den Kalender der französischen Republik in die julianische Tageszählung.

```
int frenchtojd(int month, int day, int year)
```

jdmonthname

jdmonthname gibt den Monatsnamen zurück.

```
string jdmonthname(int julianday, int mode)
```

jddayofweek

jddayofweek gibt den Wochentag zurück.

```
mixed jddayofweek(int julianday, int mode)
```

easter_date

easter_date ermittelt den Zeitstempel des Ostersonntags 00:00 Uhr für das angegebene Jahr.

```
int easter_date(int year)
```

easter_days

easter_days gibt die Anzahl der Tage zwischen Ostern und dem 21. März des angegebenen Jahres zurück.

```
int easter_days(int year)
```

Mathematische Funktionen

abs

abs gibt den absoluten Betrag zurück; der Datentyp bleibt unverändert.

```
mixed abs(mixed number)
```

acos

acos ist der Arcus Kosinus. Das Argument ist im Bogenmaß anzugeben.

```
float acos(float argument)
```

asin

asin ist der Arcus Sinus. Das Argument ist in Radiant anzugeben.

```
float asin(float argument)
```

atan

atan ist der Arcus Tangens. Das Argument ist in Radiant anzugeben.

```
float atan(float argument)
```

atan2

atan2 ist der Arcus Tangens zwischen zwei Werten.

```
float atan2(float yarg, float xarg)
```

base_convert

base_convert konvertiert Zahlenbasen.

```
string base_convert(string number, int from, int to)
```

bindec

bindec konvertiert Binärzahlen nach Dezimal.

```
int bindec(string binary)
```

ceil

ceil liefert die nächste größere ganzzahlige Zahl, rundet also praktisch immer auf.

```
int ceil(float number)
```

cos

Kosinus-Funktion, die ein Argument in Radiant erwartet.

```
float cos(float argument)
```

decbin

decbin konvertiert eine Dezimalzahl in das Binärformat.

```
string decbin(int number)
```

dechex

dechex konvertiert eine Dezimalzahl in das Hexadezimalformat.

```
string dechex(int number)
```

decoct

decoct konvertiert eine Dezimalzahl in das Oktalformat.

```
string decoct(int number)
```

deg2rad

Diese Funktion wandelt einen Winkel von Grad in Radiant um.

```
double deg2rad(double number)
```

exp

exp berechnet Potenzen der Eulerschen Zahl e.

```
float exp(float argument)
```

floor

floor rundet immer ab.

```
int floor(float number)
```

getrandmax

getrandmax gibt die größtmögliche Zufallszahl zurück.

```
int getrandmax(void)
```

hexdec

hexdec konvertiert eine Hexadezimalzahl in eine Dezimalzahl.

```
int hexdec(string hex)
```

lcg_value

Die Funktion liefert einen Pseudo-Zufallswert nach dem Verfahren des linearen kongruenten Generators (lcg).

```
float lcg_value(void)
```

log

log ermittelt den natürlichen Logarithmus.

```
float log(float argument)
```

log10

log10 ermittelt den Logarithmus zur Basis 10.

```
float log10(float argument)
```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

max

max findet den größten Wert einer variablen Anzahl Parameter.

```
mixed max(mixed arg1, mixed arg2 [, mixed argn] [, ...])
```

min

min ermittelt den niedrigsten Wert einer variablen Anzahl Parameter.

```
mixed min(mixed arg1, mixed arg2 [, mixed argn] [, ...])
```

mt_rand

mt_rand erzeugt einen besseren Zufallswert.

```
int mt_rand([int min] [, int max])
```

mt_srand

mt_srand setzt den Startwert für den mt_rand-Zufallsgenerator.

```
void mt_srand(int seedvalue)
```

mt_getrandmax

mt_getrandmax zeigt den größtmöglichen Zufallszahlenwert der mt-Funktionen an.

```
int mt_getrandmax(void)
```

number_format

number_format formatiert Zahlen als Zeichenkette.

```
string number_format(float number,  
                      [[int decimals], [string dec_point], [string thousand_sep]])
```

octdec

octdec konvertiert eine Oktalzahl in das Dezimalformat.

```
int octdec(string octal)
```

pi

pi gibt die Zahl Pi zurück.

```
double pi(void)
```

pow

pow berechnet eine Potenz.

```
float pow(float base, float exponent)
```

rad2deg

Die Funktion rad2deg wandelt einen Winkel in Radiant in Grad um.

```
double rad2deg(double number)
```

rand

rand erzeugt eine Zufallszahl.

```
int rand([int min] [, int max])
```

round

round rundet mathematisch korrekt auf ganze Zahlen.

```
double round(double value [, int precision])
```

sin

Ermittelt den Sinus für ein in Radiant gegebenes Argument.

```
float sin(float argument)
```

sqrt

sqrt berechnet die Quadratwurzel.

```
float sqrt(float argument)
```

srand

srand setzt den Startwert des Zufallszahlengenerators.

```
void srand(int seed)
```

tan

tan ermittelt den Tangens für das in Radiant übergebene Argument.

```
float tan(float argument)
```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Funktionen beliebiger Genauigkeit

bcadd

bcadd addiert zwei Zahlen beliebiger Genauigkeit. Der optionale Parameter *scale* gibt die Anzahl der Nachkommastellen an.

```
string bcadd(string left, string right [, int scale])
```

bccomp

bccomp vergleicht zwei Zahlen beliebiger Genauigkeit.

```
int bccomp(string left, string right [, int scale])
```

bcdiv

bcdiv dividiert zwei Zahlen beliebiger Genauigkeit.

```
string bcdiv(string left, string right [, int scale])
```

bcmod

bcmod ermittelt den Rest einer Ganzzahldivision.

```
string bcmod(string left, string modulus)
```

bcmul

bcmul multipliziert zwei Zahlen beliebiger Genauigkeit.

```
string bcmul(string left, string right [, int scale])
```

bcpow

bcpow potenziert eine Zahl *x* mit *y*.

```
string bcpow(string x, string y [, int scale])
```

bcscale

bcscale setzt die in den anderen Funktionen optionale Angabe der Genauigkeit auf einen festen Wert.

```
string bcscale(int scale)
```

bcsqrt

bcsqrt berechnet die Quadratwurzel. Der optionale Parameter *scale* gibt die Anzahl der Nachkommastellen an.

```
string bcsqrt(string operand [, int scale])
```

bcsub

bcsub subtrahiert zwei Zahlen beliebiger Genauigkeit.

```
string bcsub(string left, string right [, int scale])
```

Funktionen für das Dateisystem

Die Funktionen für das Dateisystem umfassen Dateien und Verzeichnisse. Die Funktionen sind aus naheliegenden Gründen hier im Gegensatz zur Originaldokumentation zusammengefasst worden.

dir

dir erstellt eine Verzeichnisklasse mit folgenden Methoden:

- read
- rewind
- close
- handle
- path

```
new dir(string directory)
```

chdir

chdir wechselt das aktuelle Verzeichnis.

```
int chdir(string directoryname)
```

closedir

closedir schließt ein mit opendir geöffnetes Verzeichnishandle.

```
void closedir(int dir_handle)
```

Ein Beispiel finden Sie bei der Funktion readdir.

getcwd

Die Methode gibt das aktuelle Verzeichnis zurück. Wenn zuvor kein Verzeichniswechsel erfolgte, ist dies das Verzeichnis, indem das Skript ausgeführt wird.

```
string getcwd(void)
```

opendir

`opendir` öffnet ein Verzeichnis und erzeugt ein Handle darauf, das andere Funktionen zum Zugriff auf das Verzeichnis nutzen.

```
int opendir(string pathname)
```

readdir

`readdir` liest einen Eintrag (Dateinamen) aus dem Verzeichnis.

```
string readdir(int dir_handle)
```

rewinddir

`rewinddir` setzt den Zeiger auf den ersten Eintrag des zuletzt gelesenen Verzeichnisses zurück.

```
void rewinddir(int dir_handle)
```

basename

`basename` gibt den Namensbestandteil der Datei in einem Pfad zurück. Allerdings prüft die Funktion nicht wirklich, ob es sich um eine Datei handelt; es wird einfach der letzte Teil des Pfades zurückgegeben.

```
string basename(string pathname)
```

chgrp

`chgrp` weist die Datei einer Benutzergruppe zu. Die Funktion hat unter Windows keinen Effekt.

```
int chgrp(string filename, mixed group)
```

chown

`chown` ändert den Eigentümer der Datei. Die Funktion hat unter Windows keinen Effekt.

```
int chown(string filename, mixed user)
```

chmod

chmod ändert die Zugriffsrechte einer Datei

```
int chmod(string filename, int mode)
```

clearstatcache

clearstatcache löscht den Zwischenspeicher mit den Dateistatuswerten. Normalweise werden diese Werte nur beim ersten Zugriff gelesen und anschließend aus einem internen Speicher geholt, um die Ausführung zu beschleunigen.

```
void clearstatcache(void)
```

copy

copy kopiert eine Datei von *sourcename* nach *destinationname*.

```
bool copy(string sourcename, string destinationname)
```

dirname

dirname gibt den Pfadanteil einer vollständigen Pfadangabe zurück. Allerdings prüft die Funktion nicht wirklich, ob es sich beim letzten Teil um eine Datei handelt, es wird einfach der erste Teil des Pfades zurückgegeben.

```
string dirname(string pathname)
```

diskfreespace

diskfreespace gibt den freien Speicherplatz eines Volumes bzw. Verzeichnisses in Byte zurück. Die Funktion berücksichtigt Disk Quotas.

```
float diskfreespace(string directoryname)
```

fclose

fclose schließt einen Dateizeiger.

```
bool fclose(int filehandle)
```

fEOF

fEOF testet den Dateizeiger auf das Dateiende. Am Dateiende wird TRUE zurückgegeben, sonst FALSE. Die Datei muss zuvor mit fopen, popen oder fsopen geöffnet worden sein.

```
int fEOF(int filehandle)
```

fflush

Die Funktion forciert die Ausgabe aller noch im Ausgabepuffer befindlichen Zeichen beim Schreiben von Dateien.

```
bool fflush(int filehandle)
```

fgetc

fgetc holt ein Zeichen von der aktuellen Position des Dateizeigers.

```
string fgetc(int filehandle)
```

fgetcsv

fgetcsv holt eine Zeile aus der Datei und zerlegt die Zeile anhand des Trennzeichens *delimiter*. Das Ergebnis wird als Array zurückgegeben.

```
array fgetcsv(int filehandle, int length [, string delimiter])
```

fgets

fgets liest eine Zeile von der aktuellen Position des Dateizeigers. Die Zeile endet am Zeilenumbruch oder bei *length* Zeichen.

```
string fgets(int filehandle, int length)
```

fgetss

fgetss liest eine Zeile und entfernt alle HTML- und PHP-Tags. Deren Inhalt bleibt aber erhalten. In *tags* kann eine Liste von Tags aufgeführt werden, die nicht entfernt werden.

```
string fgetss(int filehandle, int length [, string tags])
```

file

file liest eine Textdatei komplett in ein Array ein. Jede Zeile ist ein Element. Das Zeilenumbruchzeichen \n bleibt erhalten.

```
array file(string filename [, int include])
```

file_exists

file_exists prüft, ob eine Datei existiert. Im Erfolgsfall wird TRUE zurückgegeben, sonst FALSE.

```
bool file_exists(string filename)
```

fileatime

fileatime ermittelt Datum und Uhrzeit des letzten Dateizugriffs.

```
int fileatime(string filename)
```

filectime

filectime ermittelt Datum und Uhrzeit der letzten Änderung des Dateizeigers Inode.

```
int filectime(string filename)
```

filegroup

filegroup ermittelt die Dateigruppenzugehörigkeit.

```
int filegroup(string filename)
```

fileinode

fileinode gibt den Dateizeiger Inode aus.

```
int fileinode(string filename)
```

filemtime

filemtime ermittelt Datum und Uhrzeit der letzten Änderung an der Datei.

```
int filemtime(string filename)
```

fileowner

fileowner ermittelt den Eigentümer der Datei.

```
int fileowner(string filename)
```

fileperms

fileperms ermittelt die Zugriffsrechte auf die Datei.

```
int fileperms(string filename)
```

filesize

filesize ermittelt die Dateigröße.

```
int filesize(string filename)
```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

filetype

filetype ermittelt den Dateityp.

```
string filetype(string filename)
```

flock

flock verriegelt eine Datei ohne Unterstützung durch das Dateisystem. Der Parameter *operation* bestimmt die Art der Verriegelung; der aktuelle Zustand wird als TRUE (Datei offen) oder FALSE (Datei gesperrt) zurückgegeben.

```
bool flock(int filehandle, int operation [, int blockmode])
```

fopen

fopen öffnet eine Datei vom lokalen Dateisystem oder über eine URL.

```
int fopen(string filename, string mode)
```

fpassthru

fpassthru gibt alle Daten von einem Dateizeiger direkt aus.

```
int fpassthru(int filehandle)
```

fputs

fputs schreibt Daten an die Position des Dateizeigers. Die Funktion ist mit fwrite identisch.

```
int fputs(int filehandle, string strvar [, int length])
```

fread

fread liest Binärdaten aus einer Datei. Die Funktion liest *length* Bytes, maximal jedoch bis zum Dateiende.

```
string fread(int filehandle, int length)
```

fscanf

Diese Funktion liest Daten aus einer Datei ein und interpretiert sie anhand der vorgegebenen Formatzeichenkette *forms*.

```
mixed fscanf(int filehandle, string forms [, string vari...])
```

fseek

fseek setzt den Dateizeiger um einen Offset *offset* vor oder zurück.

```
int fseek(int filehandle, int offset)
```

fstat

fstat ermittelt verschiedene Dateiinformationen und gibt diese in einem Array zurück. Angegeben wird ein Dateihandle auf eine offene Datei.

```
array fstat(string filehandle)
```

ftell

ftell ermittelt die aktuelle Position des Dateizeigers.

```
int ftell(int filehandle)
```

ftruncate

Die Funktion entfernt alle Zeichen in einer Datei ab der angegebenen Position *size*. Die Datei muss zuvor im Schreibmodus im Zugriff sein.

```
int ftruncate(int filehandle, int size)
```

fwrite

fwrite schreibt Binärdaten in eine Datei.

```
int fwrite(int filehandle, string strvar [, int length])
```

set_file_buffer

set_file_buffer setzt die Zeichenpufferung für eine Datei auf die Größe *buffer*.

```
int set_file_buffer(int filehandle, int buffer)
```

is_dir

is_dir ermittelt, ob ein Dateiname ein Verzeichnis ist, und gibt dann TRUE zurück.

```
bool is_dir(string dirname)
```

is_executable

is_executable ermittelt, ob die Datei ausführbar ist und gibt dann TRUE zurück.

```
bool is_executable(string filename)
```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

is_file

`is_file` ermittelt, ob der Name eine Datei ist. Existiert die Datei, gibt die Funktion `TRUE` zurück.

```
bool is_file(string filename)
```

is_link

`is_link` ermittelt, ob die Datei ein symbolischer Link ist, und gibt dann `TRUE` zurück.

```
bool is_link(string filename)
```

is_readable

`is_readable` ermittelt, ob die Datei lesbar ist, existiert und gibt dann `TRUE` zurück.

```
bool is_readable(string filename)
```

is_writeable

`is_writeable` ermittelt, ob in die Datei geschrieben werden kann, und gibt dann `TRUE` zurück.

```
bool is_writeable(string filename)
```

is_uploaded_file

Diese Funktion gibt `TRUE` zurück, wenn die durch *file* bezeichnete Datei durch ein HTTP-Upload hochgeladen wurde.

```
bool is_uploaded_file(string file)
```

link

`link` erzeugt einen absoluten Link *link* mit dem Ziel *target*.

```
int link(string target, string link)
```

linkinfo

`linkinfo` ermittelt Informationen über den Link *name*.

```
int linkinfo(string name)
```

mkdir

`mkdir` erzeugt ein Verzeichnis mit dem Namen *folder*. Der Parameter *mode* gibt die Zugriffsrechte an.

```
bool mkdir(string folder, int mode)
```

move_uploaded_file

Diese Funktion verschiebt hochgeladene Dateien nach einem HTTP-Upload. Die Funktion gibt FALSE zurück, wenn *file* nicht existiert, sonst TRUE.

```
bool move_uploaded_file(string file, string targetdir)
```

pclose

pclose schließt ein Prozessdateihandle. Wenn ein Prozess verbunden ist und läuft, wird er terminiert.

```
int pclose(int filehandle)
```

popen

popen öffnet den Prozesszeiger.

```
int popen(string command, string mode)
```

readfile

readfile gibt eine Datei zum Browser aus und lässt das Dateihandle offen.

```
int readfile(string filename)
```

readlink

readlink ermittelt das Ziel eines symbolischen Links *name*.

```
string readlink(string name)
```

rename

rename benennt eine Datei *oldname* in *newname* um.

```
bool rename(string oldname, string newname)
```

rewind

rewind setzt den Dateizeiger auf das erste Byte der Datei.

```
bool rewind(int filehandle)
```

rmdir

rmdir entfernt ein Verzeichnis mit dem Namen *directory*.

```
bool rmdir(string directory)
```

stat

stat ermittelt verschiedene Dateiinformationen und gibt diese in einem Array zurück.

```
array stat(string filename)
```

lstat

lstat ermittelt Informationen über symbolische Links und gibt sie als Array zurück.

```
array lstat(string filename)
```

realpath

Gibt den wirklichen Pfad zu einer Datei zurück. Je nach System entspricht dies der internen Darstellung. Wenn der Name nicht existiert, wird eine leere Zeichenkette zurückgegeben. Alle relativen Pfadangaben wie »..« oder »..
« werden aufgelöst.

```
string realpath(string filename)
```

symlink

symlink erzeugt einen symbolischen Link. Symbolische Links sind unter Windows nicht verfügbar.

```
int symlink(string target, string linkname)
```

tempnam

tempnam erzeugt eine eindeutige und einmalige temporäre Datei und gibt deren Namen zurück. Die Datei verbleibt allerdings physisch auf der Festplatte und sollte nach Gebrauch wieder gelöscht werden. Namen gelöschter Dateien können wiederverwendet werden.

```
string tempnam(string dirname, string prefix)
```

tmpfile

Diese Funktion erzeugt eine temporäre Datei mit Schreibrechten und gibt ein Handle darauf zurück. Diese Datei kann dann mit allen Dateifunktionen, die Handle akzeptieren, genutzt werden.

```
int tmpfile(void)
```

touch

touch setzt das Datum der letzten Änderung einer Datei. Wenn die Datei nicht existiert, wird sie erzeugt.

```
int touch(string filename, int timestamp)
```

umask

umask ändert die aktuelle Umask (Zugriffsrechte) oder gibt die aktuellen Einstellungen zurück.

```
int umask(int mask)
```

unlink

unlink löscht eine Datei.

```
int unlink(string filename)
```

Kompressionsfunktionen

gzclose

Schließt das Verbindungshandle zu einer komprimierten Datei. *ziphandle* verweist auf eine zuvor mit *gzopen* geöffnete Datei.

```
int gzclose(int ziphandle)
```

gzcompress

Die Funktion komprimiert die als Zeichenkette übergebenene Daten und gibt das Komprimat zurück. *level* gibt das Kompressionsniveau an und kann zwischen 0 und 9 sein. 0 komprimiert nicht, 9 maximal.

```
string gzcompress(string uncompresseddata [, int level])
```

gzeof

Mit dieser Funktion wird getestet, ob in der aktuellen komprimierten Datei das Dateiende erreicht wurde oder ein Fehler aufgetreten ist. In diesen Fällen wird TRUE zurückgegeben.

```
bool gzeof(int ziphandle)
```

gzfile

Diese Funktion liest eine komprimierte Datei in ein Array ein. Die Datei wird dabei *nicht* dekomprimiert. Wenn der Parameter *useinclude* TRUE ist, wird der in der PHP.INI eingestellte Standardpfad zuerst durchsucht.

```
array gzfile(string file [, int useinclude])
```

gzgetc

Die Funktion liest ein unkomprimiertes Zeichen aus einer der durch *ziphandle* adressierten Datei.

```
string gzgetc(int ziphandle)
```

gzgets

Die Funktion liest eine Zeile unkomprimierter Zeichen aus einer der durch *ziphandle* adressierten Datei.

```
string gzgetc(int ziphandle [, int length])
```

gzgetss

Die Funktion liest eine Zeile unkomprimierter Zeichen aus einer der durch *ziphandle* adressierten Datei. Außerdem werden alle HTML-Tags aus der Zeichenkette entfernt. Der Parameter *tagsallowed* nennt eine durch Kommata getrennte Liste von Tags, die nicht entfernt werden sollen.

```
string gzgetss(int ziphandle [, int length] [, string tagsallowed])
```

gzopen

Öffnet eine Datei oder legt eine neue an, in die komprimierte Daten geschrieben werden. Die Funktion gibt das Handle *ziphandle* zurück.

```
int gzopen(string file, string type [, int useinclude])
```

gzpassthru

Diese Funktion liest den Rest der Datei *ziphandle* und gibt alles unverändert an den Browser aus.

```
int gzpassthru(int ziphandle)
```

gzputs

Mit dieser Funktion werden *length* Daten aus *data* in eine GZ-Datei geschrieben, die durch *ziphandle* adressiert wird.

```
int gzputs(int ziphandle, string data [, int length])
```

gzread

Liest eine *length* Bytes aus der Datei *ziphandle* ohne Dekomprimierung ein.

```
int gzread(int ziphandle [, int length])
```

gzrewind

Setzt den Dateizeiger *ziphandle* wieder an den Anfang der Datei.

```
int gzrewind(int ziphandle)
```

gzseek

Die Funktion verschiebt den Zeiger in der durch *ziphandle* adressierten Datei um eine bestimmte Anzahl Bytes. Die Funktion kann sehr langsam sein. Bei zum Schreiben geöffneten Dateien kann nur vorwärts gesucht werden.

```
int gzseek(int ziphandle, int offset)
```

gztell

Ermittelt die Position des Dateizeigers in Bytes vom Anfang der Datei.

```
int gztell(int ziphandle)
```

gzwrite

Mit dieser Funktion werden Daten in eine Datei geschrieben. Binärwerte können problemlos genutzt werden. Der Parameter *length* zählt die Länge in unkomprimierter Form.

```
int gzwrite(int ziphandle, string data [, int length])
```

readgzfile

Die Funktion `readgzfile` liest eine komprimierte Datei ein, dekomprimiert sie und gibt sie sofort an den Browser aus.

```
int readgzfile(string file [, int useinclude])
```

gzuncompress

Die Funktion dekomprimiert eine Zeichenkette, die Daten in komprimierter Form vorliegen hat.

```
string guncompress(string compresseddata [, int maxlength])
```

Sitzungssteuerung (Sessions)

session_get_cookie_params

Diese Funktion liest die aktuellen Parameter für Cookies aus und gibt sie in einem assoziativen Array zurück.

```
array session_get_cookie_params(void)
```

session_set_cookie_params

Die Funktion setzt die aktuellen Parameter für Cookies. Diese Einstellung überschreibt die Werte in der PHP.INI nur für Laufzeit des aktuellen Skripts.

```
array session_set_cookie_params(int time [, string path] [, string domain])
```

session_decode

Die Funktion dekodiert die in der Session in Form einer Zeichenkette gespeicherten Variablen, die mit session_encode verpackt wurden.

```
bool session_decode(string data)
```

session_destroy

Löscht alle Session-Daten. Das zugehörige Cookie wird jedoch nicht sofort gelöscht. Die angelegte temporäre Datei wird sofort gelöscht. Im Fehlerfall wird FALSE zurückgegeben, sonst TRUE.

```
bool session_destroy(void)
```

session_encode

Die Funktion kodiert alle Daten der aktuellen Session in Form einer Zeichenkette.

```
string session_encode(void)
```

session_id

Für die angegebene Session (Session-Name) wird die aktuelle Session-ID ermittelt und zurückgegeben.

```
string session_id([string sessionid])
```

session_is_registered

Die Funktion prüft, ob eine Variable bereits registriert ist. Als Argument wird der Name der Variablen als Zeichenkette übergeben, nicht die Variable selbst.

```
bool session_is_registered(string varname)
```

session_module_name

Setzt oder ermittelt das Modul, mit dem die Session-Informationen verarbeitet werden.

```
string session_module_name([string modulename])
```

session_name

Ermittelt oder setzt den Namen der Session. Dieser Wert wird auch als Name des Session-Cookies verwendet, wenn mit Cookies gearbeitet wird oder als Name der GET- bzw. POST-Parameter.

```
string session_name([string sessionname])
```

session_register

Die Funktion registriert Variablen für eine Sitzung. Als Argument wird der Name der Variablen angegeben, nicht die Variable selbst.

```
bool session_register(string name [, string ...])
```

session_save_path

Die Funktion setzt oder ermittelt den Pfad, unter dem die Session-Daten abgespeichert werden.

```
string session_save_path([string path])
```

session_start

Diese Funktion startet eine neue Session und gibt immer TRUE zurück. Dabei wird eine neue Session-ID erzeugt.

```
bool session_start(void)
```

session_unregister

Die Funktion hebt die Registrierung einer Variablen wieder auf. Die Variable selbst wird im aktiven Skript nicht gelöscht.

```
bool session_unregister(string varname)
```

session_unset

Die Funktion gibt alle Sessionvariablen frei.

```
void session_unset(void)
```

session_set_save_handler

Diese Funktion registriert nutzerdefinierte Funktionen, die als Routinen für die Verwaltung genutzt werden. Damit kann anstatt Dateien auch eine MySQL-Datenbank verwendet werden. Die Implementierung müssen Sie aber selbst vornehmen. Die Parameter erwarten Namen von Funktionen.

```
void session_set_save_handler(string open, string close,  
                              string read, string write,  
                              string destroy, string gc)
```

session_cache_limiter

Liest oder setzt die aktuellen Einstellungen für die Cachesteuerung im Browser. Dieser Wert muss vor session_start gesetzt werden und ist nur für die Laufzeit des Skripts gültig.

```
string session_cache_limiter([string limiter])
```

Netzwerkzugriff

fsockopen

fsockopen öffnet eine Socketverbindung unter Unix oder im Internet.

```
int fsockopen(string hostname, int portnumber  
              [, int errnumber] [, string errstring] [, double timeout])
```

ip2long

Die Funktion wandelt eine IPv4-IP-Adresse im Byte-Format in einen numerischen Wert.

```
int ip2long(string ip_address)
```

long2ip

Die Funktion wandelt einen numerischen Wert in die korrespondierende IP-Adresse im Format »AAA.BBB.CCC.DDD« um.

```
string long2ip(int numeric_address)
```

pfssockopen

pfssockopen öffnet eine persistente Socketverbindung.

```
int pfssockopen(string hostname, int port  
                [,int errno] [, string errstr] [,int timeout])
```

socket_set_blocking

`socket_set_blocking` setzt den Blockmode. Der Blockmode beeinflusst die Arbeitsweise der Funktion `fgets`.

```
int set_socket_blocking(int handle, int blockmode)
```

socket_get_status

`socket_get_status` ermittelt Informationen über die aktuelle Verbindung.

```
array socket_get_status(int handle)
```

gethostbyaddr

`gethostbyaddr` ermittelt den Namen eines Hosts anhand der IP-Adresse, die als Zeichenkette übergeben wird.

```
string gethostbyaddr(string ip_address)
```

gethostbyname

`gethostbyname` ermittelt die IP-Adresse anhand eines Hostnamens.

```
string gethostbyname(string hostname)
```

gethostbyname1

`gethostbyname1` ermittelt alle IP-Adressen zu einem Hostnamen und gibt diese als Array zurück.

```
array gethostbyname1(string hostname)
```

checkdnsrr

`checkdnsrr` prüft einen bestimmten, durch `dnsrecord` ausgewählten DNS-Eintrag.

```
int checkdnsrr(string host [, string dnsrecord])
```

getmxrr

`getmxrr` prüft den MX-Eintrag im DNS für einen Hostnamen.

```
int getmxrr(string host, array getmxhosts [, array getprio])
```

getprotobyname

Die Funktion gibt eine Protokollnummer für einen gegebenen Protokollnamen zurück.

```
int getprotobyname(string protocol)
```

getprotobyname

Die Funktion gibt den Namen eines Protokolls anhand seiner Protokollnummer zurück.

```
string getprotobyname(int protocolnumber)
```

getservbyname

Die Funktion gibt eine Portnummer für einen gegebenen Dienst zurück.

```
int getservbyname(string service, string protocol)
```

getservbyport

Die Funktion gibt den Namen eines Dienstes anhand seiner Portnummer zurück.

```
string getservbyport(int portnumber)
```

Client-URL Funktionen

curl_close

Schließt eine CURL-Sitzung und verwirft die eingestellten Parameter.

```
void curl_close(int curlhandle)
```

curl_errno

Gibt die letzte Fehlernummer zurück. Wenn kein Fehler auftrat, wird 0 zurückgegeben.

```
int curl_errno(int curlhandle)
```

curl_error

Die Funktion gibt den Text der letzten Fehlermeldung zurück.

```
string curl_error(int curlhandle)
```

curl_exec

Führt die Abfrage des Servers mit den voreingestellten Parametern aus. Wenn die Aktion erfolgreich verlief, wird TRUE zurückgegeben.

```
bool curl_exec(int curlhandle)
```

curl_getinfo

Mit dieser Funktion werden Informationen über den letzten Transfer ermittelt.

```
array curl_getinfo(int curlhandle)
```

curl_init

Diese Funktion initialisiert eine CURL-Sitzung und gibt ein Handle zurück, das die anderen CURL-Funktionen benötigen. Es wird dort als *curlhandle* bezeichnet.

```
int curl_init([string url])
```

curl_setopt

Die Funktion setzt Optionen, die den bevorstehenden Zugriff auf eine URL näher beschreiben.

```
bool curl_setopt(int curlhandle, int option, mixed value)
```

curl_version

Die Funktion gibt die Version der verwendeten CURL-Bibliothek aus.

```
string curl_version(void)
```

FTP-Funktionen

ftp_connect

Die Funktion `ftp_connect` verbindet mit einem FTP-Server und gibt ein Handle auf diese Verbindung zurück, das von allen folgenden FTP-Funktionen genutzt wird.

```
int ftp_connect(string hostname [, int portnumber])
```

ftp_exec

Die Funktion führt ein Kommando auf dem FTP-Server aus, wenn dieser das unterstützt.

```
boolean ftp_exec(int ftphdl, string command)
```

ftp_login

Diese Funktion führt eine Anmeldung an einem FTP-Server aus.

```
int ftp_login(int ftphdl, string username, string password)
```

ftp_pwd

`ftp_pwd` gibt den Namen des aktuellen Verzeichnisses zurück.

```
mixed ftp_pwd(int ftphdl)
```

ftp_chdir

ftp_chdir wechselt das aktuelle Verzeichnis auf dem Server.

```
int ftp_chdir(int ftphdl, string newdir)
```

ftp_cdup

ftp_cdup wechselt eine Verzeichnisebene höher.

```
int ftp_cdup(int ftphdl)
```

ftp_mkdir

ftp_mkdir erzeugt ein Verzeichnis *directory*. Der Name muss nicht mit einem Schrägstrich beginnen.

```
int ftp_mkdir(int ftphdl, string directory)
```

ftp_rmdir

ftp_rmdir löscht das Verzeichnis *directory*. Das zu löschende Verzeichnis muss leer sein. Es darf nicht mehr das aktuelle Verzeichnis sein. Der Name muss mit einem Schrägstrich beginnen.

```
int ftp_rmdir(int ftphdl, string directory)
```

ftp_nlist

ftp_nlist listet die Dateien des Verzeichnisses *directory* auf. Der Name muss mit einem Schrägstrich beginnen.

```
array ftp_nlist(int ftphdl, string directory)
```

ftp_rawlist

ftp_rawlist gibt eine detaillierte Liste von Dateien im Verzeichnis *directory* zurück.

```
array ftp_rawlist(int ftphdl, string directory)
```

ftp_site

Diese Funktion sendet ein FTP-Kommando an den Server. Die möglichen Kommandos hängen vom Typ des FTP-Servers ab.

```
bool ftp_site(int ftphdl, string command)
```

ftp_systype

`ftp_systype` erkennt den Systemtyp des FTP-Servers.

```
string ftp_systype(int ftphdl)
```

ftp_pasv

`ftp_pasv` schaltet den passiven Modus *pasvmode* ein (TRUE) oder aus (FALSE).

```
int ftp_pasv(int fthdl, int pasvmode)
```

ftp_get

`ftp_get` lädt eine Datei *remotename* aus dem aktuellen Verzeichnis des FTP-Servers auf das lokale System, auf dem das PHP-Skript läuft – dies kann ebenso ein Server sein.

```
int ftp_get(int ftphdl, string localname, string remotename, int mode)
```

ftp_fget

`ftp_fget` lädt eine Datei vom FTP-Server in eine lokale geöffnete Datei, deren Handle mit *filehdl* angegeben werden muss.

```
int ftp_fget(int ftphdl, int filehdl, string remotename, int mode)
```

ftp_put

`ftp_put` überträgt eine Datei auf den FTP-Server.

```
int ftp_put(int ftphdl, string remotename, string localname, int mode)
```

ftp_fput

`ftp_fput` überträgt eine lokal geöffnete Datei auf den FTP-Server.

```
int ftp_fput(int ftphdl, string remotename, int filehdl, int mode)
```

ftp_size

`ftp_size` gibt die Größe der Datei *pathname* zurück.

```
int ftp_size(int ftphdl, string pathname)
```

ftp_mdtm

`ftp_mdtm` gibt das Datum der letzten Änderung einer Datei als Unix-Zeitstempel zurück.

```
int ftp_mdtm(int ftphdl, string pathname)
```

ftp_rename

`ftp_rename` benennt die Datei *fromname* in *toname* um. Wenn sich die Pfade unterscheiden, wird die Datei umbenannt und in das neue Verzeichnis verschoben.

```
int ftp_rename(int ftphdl, string fromname, string toname)
```

ftp_delete

`ftp_delete` löscht eine Datei mit dem Namen *name*, dieser Name kann auch eine Pfadangabe beinhalten.

```
int ftp_delete(int ftphdl, string name)
```

ftp_quit

`ftp_quit` beendet die Verbindung zu einem FTP-Server.

```
int ftp_quit(int ftphdl)
```

Verschlüsselungs-Funktionen

mcrypt_get_cipher_name

`mcrypt_get_cipher_name` ermittelt den Namen der Chiffre. Wenn die Chiffre nicht existiert, wird FALSE zurückgegeben.

```
string mcrypt_get_cipher_name(int cipher)
```

mcrypt_get_block_size

`mcrypt_get_block_size` ermittelt die Blockgröße der Chiffre *cipher* in Bytes.

```
int mcrypt_get_block_size(int cipher)
```

mcrypt_get_key_size

`mcrypt_get_key_size` ermittelt die Schlüssellänge des Schlüssels *cipher* in Bytes.

```
int mcrypt_get_key_size(int cipher)
```

mcrypt_create_iv

`mcrypt_create_iv` ermittelt den Initialisierungsvektor auf der Basis einer zufälligen Quelle *source*.

```
string mcrypt_create_iv(int size, int source)
```

mcrypt_cbc

mcrypt_cbc ver- und entschlüsselt im CBC-Mode.

```
int mcrypt_cbc(int cipher, string key, string data, int mode [, string iv])
```

mcrypt_cfb

mcrypt_cfb ver- und entschlüsselt im CFB-Mode.

```
int mcrypt_cfb(int cipher, string key, string data, int mode, string iv)
```

mcrypt_ecb

mcrypt_ecb ver- und entschlüsselt im ECB-Mode.

```
int mcrypt_ecb(int cipher, string key, string data, int mode)
```

mcrypt_ofb

mcrypt_ofb ver- und entschlüsselt im OFB-Mode.

```
int mcrypt_ofb(int cipher, string key, string data, int mode, string iv)
```

mcrypt_list_algorithms

Die Funktion erzeugt ein Array mit allen unterstützten Codierungsverfahren.

```
array mcrypt_list_algorithms([string pathtolib])
```

mcrypt_list_modes

Diese Funktion erzeugt ein Array mit allen unterstützten Verschlüsselungsmodi.

```
array mcrypt_list_modes([string pathtolib])
```

mcrypt_get_iv_size

Gibt die Größe des Initialisierungsvektors einer bestimmten Kombination von Verschlüsselungsverfahren und -modus zurück.

```
int mcrypt_get_iv_size(string cipher, string mode)
```

mcrypt_encrypt

mcrypt_encrypt verschlüsselt Text nach dem angegebenen Verfahren und mit den entsprechenden Parametern.

```
string mcrypt_encrypt(string cipher, string key,  
                      string data, string mode [, string iv])
```

mcrypt_decrypt

`mcrypt_decrypt` verschlüsselt Text nach dem angegebenen Verfahren und mit den entsprechenden Parametern.

```
string mcrypt_decrypt(string cipher, string key,  
                      string cdata, string mode [, string iv])
```

mcrypt_module_open

Diese Funktion öffnet eine Bibliothek mit Verschlüsselungsfunktionen und gibt ein Handle auf diese Ressource zurück. In den nachfolgenden Funktionen dieses Abschnitts wird dieses Handle als Parameter *handle* bezeichnet.

```
int mcrypt_module_open(string algorithm, string algorithm_directory,  
                      string mode, string mode_directory)
```

mcrypt_generic_init

Diese Funktion initialisiert einen Puffer für die Verschlüsselungsoperationen.

```
int mcrypt_generic_init(int handle, string key, string iv)
```

mcrypt_generic

Diese Funktion verschlüsselt Daten auf der Basis einer Verschlüsselungsbibliothek. Zum Zugriff darauf wird das Handle *handle* verwendet.

```
string mcrypt_generic(int handle, string data)
```

mdecrypt_generic

Diese Funktion verschlüsselt Daten auf der Basis einer Verschlüsselungsbibliothek. Zum Zugriff darauf wird das Handle *handle* verwendet.

```
string mdecrypt_generic(int handle, string cryptdata)
```

mcrypt_generic_end

Die Funktion schließt ein Verschlüsselungsmodul und leert alle verwendeten Puffer.

```
bool mcrypt_generic_end(int handle)
```

mcrypt_enc_self_test

Diese Funktion testet das Modul und gibt TRUE zurück, wenn es erwartungsgemäß funktioniert, sonst FALSE.

```
bool mcrypt_enc_self_test(int handle)
```

mcrypt_enc_is_block_algorithm_mode

Mit dieser Funktion kann festgestellt werden, ob das geöffnete Verschlüsselungsmodul im Blockmodus arbeitet oder nicht. Nur für den Modus stream ist das normalerweise nicht der Fall.

```
int mcrypt_enc_is_block_algorithm_mode(int handle)
```

mcrypt_enc_is_block_algorithm

Mit dieser Funktion kann festgestellt werden, ob das geöffnete Verschlüsselungsmodul einen Blockalgorithmus verwendet oder nicht.

```
int mcrypt_enc_is_block_algorithm(int handle)
```

mcrypt_enc_is_block_mode

Mit dieser Funktion kann festgestellt werden, ob das geöffnete Verschlüsselungsmodul Daten blockweise ausgibt oder nicht.

```
int mcrypt_enc_is_block_mode(int handle)
```

mcrypt_enc_get_block_size

Diese Funktion ermittelt die Größe der Blöcke beim aktuellen Verschlüsselungsmodul, wenn dieser im Blockmodus arbeitet.

```
int mcrypt_enc_get_block_size(int handle)
```

mcrypt_enc_get_key_size

Diese Funktion ermittelt die maximale Schlüsselgröße, die das angegebene Verschlüsselungsmodul verwendet.

```
int mcrypt_enc_get_key_size(int handle)
```

mcrypt_enc_get_supported_key_sizes

Diese Funktion gibt ein Array zurück, das alle Schlüsselgrößen enthält, die das verwendete Verschlüsselungsmodul verarbeiten kann.

```
array mcrypt_enc_get_supported_key_sizes(int handle)
```

mcrypt_enc_get_iv_size

Die Funktion gibt die Größe des Initialisierungsvektors für das verwendete Verschlüsselungsmodul zurück.

```
int mcrypt_enc_get_iv_size(int handle)
```

mcrypt_enc_get_algorithms_name

Mit dieser Funktion ermitteln Sie den Namen des Algorithmus des verwendeten Verschlüsselungsmoduls.

```
string mcrypt_enc_get_algorithms_name(int handle)
```

mcrypt_enc_get_modes_name

Gibt den Namen des Modus zurück, indem das verwendete Verschlüsselungsmodul arbeitet.

```
string mcrypt_enc_get_modes_name(int handle)
```

mcrypt_module_self_test

Diese Funktion führt einen Selbsttest aus. Dabei wird die aktuelle Bibliothek oder die im Pfad *library* gefundene verwendet.

```
bool mcrypt_module_self_test(string algorithm [, string library])
```

mcrypt_is_block_algorithm_mode

Mit dieser Funktion kann festgestellt werden, ob das geöffnete Verschlüsselungsmodul im Blockmodus arbeitet oder nicht. Nur für den Modus stream ist das normalerweise nicht der Fall.

```
int mcrypt_is_block_algorithm_mode(string algorithm [, string library])
```

mcrypt_module_is_block_algorithm

Mit dieser Funktion kann festgestellt werden, ob das geöffnete Verschlüsselungsmodul eine Blockalgorithmus verwendet oder nicht.

```
int mcrypt_module_is_block_algorithm(string algorithm [, string library])
```

mcrypt_module_is_block_mode

Mit dieser Funktion kann festgestellt werden, ob das geöffnete Verschlüsselungsmodul Daten blockweise ausgibt oder nicht.

```
int mcrypt_module_is_block_mode(string algorithm [, string library])
```

mcrypt_module_get_algo_block_size

Die Funktion ermittelt die Blockgröße, die ein bestimmter Algorithmus des angegebenen Moduls verwendet.

```
int mcrypt_module_get_algo_block_size(string algorithm [, string library])
```

mcrypt_module_get_algo_key_size

Diese Funktion gibt die maximal unterstützte Schlüsselgröße des angegebenen Algorithmus zurück.

```
int mcrypt_module_get_algo_key_size(string algorithm [, string library])
```

mcrypt_module_get_algo_supported_key_sizes

Diese Funktion gibt ein Array mit allen unterstützten Schlüsselgrößen zurück.

```
array mcrypt_module_get_algo_supported_key_sizes(string algorithm [, string lib])
```

Hashfunktionen

mhash_get_hash_name

Die Funktion `mhash_get_hash_name` ermittelt den Namen des spezifischen Hashes *hashmode*.

```
string mhash_get_hash_name(int hashmode)
```

mhash_get_block_size

`mhash_get_block_size` ermittelt die Blockgröße des Hashes in Bytes. Wenn der Hash nicht existiert, wird FALSE zurückgegeben.

```
int mhash_get_block_size(int hashmode)
```

mhash_count

`mhash_count` ermittelt die größte ID, für die Hashmodi verfügbar sind.

```
int mhash_count(void)
```

mhash

`mhash` berechnet einen Hash auf Grundlage der Daten *data*.

```
string mhash(int hashmode, string data)
```

mhash_keygen_s2k

Die Funktion erzeugt einen Schlüssel auf der Grundlage des Kennwortes *password* und der Zeichenkette *salt*. Technisch basiert das Verfahren auf der in RCF 2440 beschriebenen OpenPGP-Methode.

```
string mhash_keygen_s2k(int hash, string password, string salt, int bytes)
```

WDDX-Funktionen

wddx_serialize_value

`wddx_serialize_value` serialisiert einen Wert in ein WDDX-Paket und gibt das Paket zurück.

```
string wddx_serialize_value(mixed var [, string comment])
```

wddx_serialize_vars

`wddx_serialize_vars` serialisiert mehrere Werte in ein WDDX-Paket und gibt das Paket zurück.

```
string wddx_serialize_vars(string var_name | array var_names [, ... ])
```

wddx_packet_start

`wddx_packet_start` startet ein neues WDDX-Paket und gibt eine ID zur späteren Verwendung zurück.

```
int wddx_packet_start([string comment])
```

wddx_packet_end

`wddx_packet_end` beendet ein WDDX-Paket, das durch die `packet_id` adressiert wird.

```
int wddx_packet_end(int packet_id)
```

wddx_add_vars

`wddx_add_vars` serialisiert Variablen und fügt diese einem Paket hinzu.

```
void wddx_add_vars(int packet_id, mixed var [, mixed vars...])
```

wddx_deserialize

`wddx_deserialize` deserialisiert ein WDDX-Paket.

```
mixed wddx_deserialize(string packet_id)
```

XML-Funktionen (Expat)

xml_parser_create

`xml_parser_create` erzeugt einen neuen XML-Parser und gibt ein Handle darauf zurück.

```
int xml_parser_create([string encoding])
```

xml_set_element_handler

`xml_set_element_handler` setzt die Start- und Ende-Ereignisbehandlungsroutinen für den Parser *parserid*.

```
int xml_set_element_handler(int parserid,  
                           string startElementHandler, string endElementHandler)
```

xml_set_character_data_handler

`xml_set_character_data_handler` setzt die Behandlungsfunktionen für Zeichendaten.

```
int xml_set_character_data_handler(int parserid, string handler)
```

xml_set_processing_instruction_handler

`xml_set_processing_instruction_handler` setzt die Behandlung von Prozessinformationen (PI), beispielsweise eingebetteten PHP-Code.

```
int xml_set_processing_instruction_handler(int parserid, string handler)
```

xml_set_default_handler

`xml_set_default_handler` setzt eine Standardbehandlungsroutine auf *handler*.

```
int xml_set_default_handler(int parserid, string handler)
```

xml_set_unparsed_entity_decl_handler

Die Funktion `xml_set_unparsed_entity_decl_handler` setzt eine Behandlungsroutine *handler* für externe ungeparste Entitäten des Parser *parser*. Parameter sind das Parser-Handle und die Handler-Funktion.

```
int xml_set_unparsed_entity_decl_handler(int parserid, string handler)
```

xml_set_notation_decl_handler

`xml_set_notation_decl_handler` setzt eine Behandlungsroutine für die Bearbeitung von Notationen.

```
int xml_set_notation_decl_handler(int parserid, string handler)
```

xml_set_external_entity_ref_handler

`xml_set_external_entity_ref_handler` setzt eine Behandlungsroutine für externe Entitäten.

```
int xml_set_external_entity_ref_handler(int parserid, string handler)
```

xml_parse

xml_parse startet das Parsen des Dokuments.

```
int xml_parse(int parserid, string data [, int isFinal])
```

xml_get_error_code

xml_get_error_code ermittelt den Fehlercode der letzten Aktion des Parsers.

```
int xml_get_error_code(int parserid)
```

xml_error_string

xml_error_string ermittelt die Fehlermeldung der letzten Aktion des Parsers.

```
string xml_error_string(int code)
```

xml_get_current_line_number

xml_get_current_line_number gibt die aktuelle Zeile zurück, wo sich der Parser befindet.

```
int xml_get_current_line_number(int parserid)
```

xml_get_current_column_number

xml_get_current_column_number gibt die aktuelle Spalte zurück, wo sich der Parser befindet.

```
int xml_get_current_column_number(int parserid)
```

xml_get_current_byte_index

xml_get_current_byte_index ermittelt den aktuellen Byte-Index (Zeichen in der Datei).

```
int xml_get_current_byte_index(int parserid)
```

xml_parser_free

xml_parser_free gibt den vom Parser verwendeten Speicher wieder frei.

```
string xml_parser_free(int parserid)
```

xml_parser_set_option

xml_parser_set_option setzt die Option *option* des Parsers auf den Wert *value*.

```
int xml_parser_set_option(int parserid, int option, mixed value)
```

xml_parser_get_option

xml_parser_get_option ermittelt die Option *option* des XML-Parsers *parser*.

```
mixed xml_parser_get_option(int parserid, int option)
```

utf8_decode

utf8_decode konvertiert UTF-8 kodierte Zeichen nach ISO-8859-1.

```
string utf8_decode(string data)
```

utf8_encode

utf8_encode kodiert ISO-8859-1 Zeichen nach UTF-8.

```
string utf8_encode(string data)
```

DOMXML-Funktionen

xmlDoc

Diese Klasse erzeugt ein Objekt, das ein XML-Dokument *xmlDocument* repräsentiert.

```
object xmlDoc(string xmlDocument)
```

xmlDocfile

Diese Klasse erzeugt ein Objekt, das ein XML-Dokument in *filename_of_xmlDoc* repräsentiert.

```
object xmlDocfile(string filename_of_xmlDoc)
```

xmltree

Diese Klasse erzeugt eine Objektstruktur, die ein XML-Dokument *xmlDocument* repräsentiert.

```
object xmltree(string xmlDocument)
```

domxml_add_root

Fügt dem Dokument ein weiteres Wurzelement hinzu und gibt das Objekt dieses neuen Knotens zurück.

```
object domxml_add_root(object doc, string name)
```

domxml_attributes

Gibt ein Array der Attribute des Knotenobjekts zurück.

```
array domxml_attributes(object node, string attributename)
```

domxml_attrname

Diese Funktion gibt eine Liste der Attribute des Dokuments zurück.

```
array domxml_attrname(object xml doc)
```

domxml_children

Gibt ein Objekt auf die untergeordneten Elemente zurück.

```
object domxml_children(object rootnode)
```

domxml_dtd

Diese Funktion gibt die DTD des Dokuments zurück.

```
object domxml_dtd(object xml doc)
```

domxml_dumpmem

Diese Funktion gibt das aktuelle XML-Dokument in lesbarer Form aus.

```
string domxml_dumpmem(object xml doc)
```

domxml_getattr

Ermittelt den Wert eines bestimmten Attributes.

```
string domxml_getattr(object node, string attributname)
```

domxml_lastchild

Liest das letzte untergeordnete Objekt eines Zweigs.

```
object domxml_lastchild(object node)
```

domxml_new_child

Erzeugt ein neues untergeordnetes Objekt unterhalb von *node*. Sie können den Parameter *content* leer lassen und später mit der Funktion `domxml_set_content` eintragen.

```
object domxml_new_child(object node, string name, string content)
```

domxml_new_xmldoc

Diese Funktion erzeugt ein neues DOM-Objekt auf der Grundlage der übergebenen Daten.

```
object domxml_new_xmldoc(string xml doc)
```

domxml_node

Erzeugt einen Verweis auf ein Knotenobjekt.

```
object domxml_node(string name)
```

domxml_parent

Die Funktion `domxml_parent` gibt einen Verweis auf das übergeordnete Objekt in der Hierarchie zurück.

```
object domxml_parent(object node)
```

domxml_root

Die Funktion gibt das Wurzelement als Objekt zurück.

```
object domxml_root(object document)
```

domxml_rootnew

Die Funktion gibt alle neuen Rootobjekte des Dokuments zurück.

```
array domxml_rootnew(object xml doc)
```

domxml_setattr

Diese Funktion setzt ein Attribut in einem Knotenelement.

```
void domxml_setattr(object node, string attribut, string value)
```

domxml_set_content

Mit dieser Funktion wird der Inhalt eines Container-Tags gesetzt.

```
void domxml_set_content(object node, string content)
```

node_attributes

Die Funktion gibt die untergeordneten Knoten mit bestimmten Attributen zurück.

```
array node_attributes(string attributes, object node)
```

node_children

Gibt eine Liste untergeordneter Knoten zurück.

```
array node_attributes(object node)
```

node_namespace

Die Funktion ermittelt den verwendeten Namensraum.

```
array node_namespace(object node)
```

PDF-Funktionen

pdf_set_info

`pdf_set_info` setzt Stamminformationen in einem PDF-Dokument.

```
void pdf_set_info(int pdfhdl, string field, string fieldvalue)
```

pdf_open

`pdf_open` öffnet ein neues PDF-Dokument. Die Funktion gibt einen Zeiger auf die Resource zurück, die alle anderen Funktionen verwenden und die dort mit *pdfhandle* bezeichnet wird.

```
int pdf_open([int filehandle])
```

pdf_close

`pdf_close` schließt ein PDF-Dokument. Sie müssen die Datei außerdem noch mit `fclose` physisch schließen, wenn sie angelegt wurde.

```
void pdf_close(int pdfhandle)
```

pdf_begin_page

`pdf_begin_page` erzeugt eine neue Seite in einem PDF-Dokument. Der Parameter *height* gibt die Höhe, *width* die Breite der Seite in Punkt an.

```
void pdf_begin_page(int pdfhandle, double height, double width)
```

pdf_end_page

`pdf_end_page` beendet eine Seite. Danach kann die Seite nicht mehr verändert werden.

```
void pdf_end_page(int pdfhandle)
```

pdf_show

pdf_show gibt Text an der aktuellen Stelle in das PDF-Dokument aus.

```
void pdf_show(int pdfhandle, string text)
```

pdf_show_boxed

pdf_show_boxed gibt Text an der aktuellen Stelle in das PDF-Dokument in einem unsichtbaren Rahmen aus.

```
void pdf_show_boxed(int pdfhandle, string text,  
                    double xpos, double ypos, double width, double height,  
                    string mode [, string feature])
```

pdf_show_xy

pdf_show_xy gibt Text an der bezeichneten Stelle in das PDF-Dokument aus.

```
void pdf_show_xy(int pdfhandle, string text, double xpos, double ypos)
```

pdf_set_font

pdf_set_font setzt den für die nächste Ausgabe verwendeten Font und die Größe.

```
void pdf_set_font(int pdfhandle, string font, double size, string encoding)
```

pdf_set_leading

pdf_set_leading setzt den Zeilenabstand bei der Textausgabe mehrerer Zeilen im Zusammenhang mit pdf_continue_text.

```
void pdf_set_leading(int pdfhandle, double distance)
```

pdf_set_parameter

Mit dieser Funktion können viele interne Parameter des Dokuments gesetzt werden.

```
void pdf_set_parameter(int pdfhandle, string type, string val)
```

pdf_get_parameter

Diese Funktion liest Parameter aus, die Zeichenketten zurückgeben.

```
string pdf_get_parameter(int pdfhandle, string type [, double modifier])
```

pdf_set_value

Mit dieser Funktion werden Einstellungen gesetzt, die numerische Werte benötigen.

```
boolean pdf_set_value(int pdfhandle, string type, mixed value)
```

pdf_get_value

Mit dieser Funktion werden Einstellungen gelesen, die numerische Werte zurückgeben.

```
double pdf_get_value(int pdfhandle, string type)
```

pdf_get_image_height

Diese Funktion ermittelt die Höhe eines eingebetteten Bildes.

```
string pdf_get_image_height(int pdfhandle, int imagehandle)
```

pdf_get_image_width

Diese Funktion ermittelt die Breite eines eingebetteten Bildes.

```
string pdf_get_image_width(int pdfhandle, int imagehandle)
```

pdf_set_text_rendering

pdf_set_text_rendering setzt die Textbehandlung für das Dokument.

```
void pdf_set_text_rendering(int pdfhandle, int mode)
```

pdf_set_horiz_scaling

pdf_set_horiz_scaling setzt das horizontale Skalieren des Textes in Prozent.

```
void pdf_set_horiz_scaling(int pdfhandle, double percent)
```

pdf_set_text_rise

pdf_set_text_rise setzt die Höhe, um die Text gegenüber der aktuellen Position angehoben werden soll.

```
void pdf_set_text_rise(int pdfhandle, double toraise)
```

pdf_set_text_pos

pdf_set_text_pos setzt die Textposition für den nächsten Aufruf von pdf_show. Dabei wird die linke untere Ecke der Grundlinie des Textfelds angegeben. Der Ursprung des Koordinatensystems der Seite beginnt ebenso links unten.

```
void pdf_set_text_pos(int pdfhandle, double xpos, double ypos)
```

pdf_set_char_spacing

Die Funktion pdf_set_char_spacing setzt den Abstand von Zeichen auf den Wert *spacing*.

```
void pdf_set_char_spacing(int pdfhandle, double spacing)
```

pdf_set_word_spacing

pdf_set_word_spacing setzt den Wortabstand im Dokument auf den Wert *spacing* fest.

```
void pdf_set_word_spacing(int pdfhandle, double spacing)
```

pdf_skew

Diese Funktion neigt das Koordinatensystem um einen bestimmten Winkel, getrennt nach x- und y-Koordinaten. Die Werte 90° und 270° sind nicht zulässig.

```
void pdf_skew(int pdfhandle, double xskew, double yskew)
```

pdf_continue_text

pdf_continue_text gibt Text in die nächste Zeile aus. Der Abstand der Zeilen wird mit pdf_set_leading gesetzt.

```
void pdf_continue_text(int pdfhandle, string text)
```

pdf_stringwidth

pdf_stringwidth ermittelt die Breite eines Textes für den aktuellen Font. Der Font muss zuvor bestimmt worden sein.

```
double pdf_stringwidth(int pdfhandle, string text)
```

pdf_save

pdf_save sichert die aktuelle Entwicklungsumgebung, ähnlich dem Postscript-Kommando GSAVE. Damit werden Änderungen an zuvor platzierten Objekten verhindert. Die Funktion sollte von pdf_restore gefolgt werden.

```
void pdf_save(int pdfhandle)
```

pdf_restore

pdf_restore holt die ursprüngliche Entwicklungsumgebung wieder zurück, ähnlich dem Postscript-Kommando GRESTORE.

```
void pdf_restore(int pdfhandle)
```

pdf_translate

pdf_translate legt die Ursprungskoordinaten fest. Damit ist der Koordinatenursprung nicht mehr links unten. Startpunkt für Linien müssen nach der Veränderung neu gesetzt werden.

```
void pdf_translate(int pdfhandle, double xpos, double ypos)
```

pdf_scale

pdf_scale setzt die Skalierung der internen Einheit. Die interne Einheit ist ein $1/72$ tel Zoll.

```
void pdf_scale(int pdfhandle, double xscale, double yscale)
```

pdf_rotate

pdf_rotate setzt einen Winkel, um den nachfolgende Objekte rotiert werden. Dies basiert auf einer Rotation des gesamten Koordinatensystems, so dass gegebenenfalls der Koordinatenursprung in das Zentrum des Objekts gelegt werden sollte, um den gewünschten Effekt zu erzielen.

```
void pdf_rotate(int pdfhandle, double angle)
```

pdf_setflat

pdf_setflat setzt die Annäherungsgenauigkeit auf einen Wert zwischen 0 und 100. Der Wert bestimmt, wie exakt Objekte am Pfad einer Vektorgrafik ausgerichtet werden. Der Standardwert ist 0.

```
void pdf_setflat(int pdfhandle, double value)
```

pdf_setlinejoin

pdf_setlinejoin setzt einen Wert der bestimmt, wie Linienecken miteinander verbunden werden. Der Wert kann 0, 1 oder 2 sein.

```
void pdf_setlinejoin(int pdfhandle, long value)
```

pdf_setlinecap

pdf_setlinecap setzt den Wert für die Bestimmung der Form der Linienenden. Der Wert kann 0, 1 oder 2 sein.

```
void pdf_setlinecap(int pdfhandle, int value)
```

pdf_setmiterlimit

pdf_setmiterlimit setzt den Parameter *value* auf einen Wert größer oder gleich 1.

```
void pdf_setmiterlimit(int pdfhandle, double value)
```

pdf_setlinewidth

pdf_setlinewidth setzt die Breite einer Linie auf den Wert *width* fest.

```
void pdf_setlinewidth(int pdfhandle, double width)
```

pdf_setdash

pdf_setdash setzt das Muster für Linien.

```
void pdf_setdash(int pdfhandle, double white, double black)
```

pdf_moveto

pdf_moveto setzt den Zeiger für die Fortsetzung der nächsten Linie auf den angegebenen absoluten Punkt im Koordinatensystem.

```
void pdf_moveto(int pdfhandle, double xpos, double ypos)
```

pdf_curveto

pdf_curveto zeichnet eine Bezier-Kurve vom aktuellen Punkt zu x_3, y_3 mit x_1, y_1 und x_2, y_2 als Kontrollpunkte.

```
void pdf_curveto(int pdfhandle, double x1, double y1,
                 double x2, double y2, double x3, double y3)
```

pdf_lineto

pdf_lineto zeichnet eine Linie vom aktuellen Punkt zu den Koordinaten $xpos$ und $ypos$.

```
void pdf_lineto(int pdfhandle, double xpos, double ypos)
```

pdf_circle

pdf_circle zeichnet einen Kreis mit dem Mittelpunkt $xpos$ und $ypos$ und dem Radius $radius$.

```
void pdf_circle(int pdfhandle, double xpos, double ypos, double radius)
```

pdf_arc

pdf_arc zeichnet einen Kreisbogen um den Mittelpunkt $xpos$ und $ypos$ und dem Radius $radius$ vom Startwinkel $start$ bis zum Endwinkel end .

```
void pdf_arc(int pdfhandle, double xpos, double ypos,
             double radius, double start, double end)
```

pdf_rect

pdf_rect zeichnet ein Rechteck von den Koordinaten $xpos$ und $ypos$ mit der Breite $width$ und der Höhe $height$.

```
void pdf_rect(int pdfhandle, double xpos, double ypos, double width, double height)
```

pdf_closepath

`pdf_closepath` schließt einen Zeichenpfad ab. Damit wird das aktuelle Ende des Pfades mit dem Anfang verbunden. Viele Funktionen wie `pdf_moveto`, `pdf_circle` und `pdf_rect` starten einen neuen Pfad. Die Anwendung ist vor allem mit Linien sinnvoll.

```
void pdf_closepath(int pdfhandle)
```

pdf_stroke

`pdf_stroke` zeichnet eine Linie an einem Pfad entlang. Der Pfad ergibt sich aus allen vorangegangenen Linienzeichenbefehlen. Ohne `pdf_stroke` würde keine Linie erscheinen.

```
void pdf_stroke(int pdfhandle)
```

pdf_closepath_stroke

`pdf_closepath_stroke` schließt den Pfad und zeichnet die zuvor bereits generierte Linie.

```
void pdf_closepath_stroke(int pdfhandle)
```

pdf_fill

`pdf_fill` füllt den aktuellen Pfad. Es wird die aktuelle Füllfarbe verwendet. Der Pfad muss geschlossen sein, sonst wird die ganze Seite gefüllt.

```
void pdf_fill(int pdfhandle)
```

pdf_fill_stroke

`pdf_fill_stroke` füllt und zeichnet den aktuellen Pfad.

```
void pdf_fill_stroke(int pdfhandle)
```

pdf_closepath_fill_stroke

`pdf_closepath_fill_stroke` füllt und zeichnet den aktuellen Pfad und schließt ihn dann.

```
void pdf_closepath_fill_stroke(int pdfhandle)
```

pdf_endpath

`pdf_endpath` beendet den aktuellen Pfad und lässt ihn offen. Sie sollten diese Funktion normalerweise nicht benötigen und statt dessen die Darstellung mit `pdf_stroke` forcieren und das Objekt mit `pdf_clip` ausschneiden.

```
void pdf_endpath(int pdfhandle)
```

pdf_clip

`pdf_clip` schneidet den aktuellen Pfad aus. Sie können die Werte zuvor mit `pdf_save` speichern und nach anderen Operationen mit `pdf_restore` wiederherstellen.

```
void pdf_clip(int pdfhandle)
```

pdf_setgray_fill

`pdf_setgray_fill` setzt die Füllfarbe auf eine Graustufe. Der Wert für die Graustufe *grayvalue* muss zwischen 0 und 1 liegen. Unterstützt werden 256 Graustufen, wobei 1 der 256 entspricht. 0 ist schwarz.

```
void pdf_setgray_fill(int pdfhandle, double grayvalue)
```

pdf_setgray_stroke

Die Funktion `pdf_setgray_stroke` setzt die Zeichenfarbe auf den Grauwert *grayvalue*. Der Wert für die Graustufe muss zwischen 0 und 1 liegen. Unterstützt werden 256 Graustufen, wobei 1 der 256 entspricht. 0 ist schwarz.

```
void pdf_setgray_stroke(int pdfhandle, double grayvalue)
```

pdf_setgray

`pdf_setgray` setzt die Zeichen- und Füllfarbe auf eine Graustufe *grayvalue*. Der Wert für die Graustufe muss zwischen 0 und 1 liegen. Unterstützt werden 256 Graustufen, wobei 1 der 256 entspricht. 0 ist schwarz.

```
void pdf_setgray(int pdfhandle, double grayvalue)
```

pdf_setrgbcolor_fill

`pdf_setrgbcolor_fill` setzt die Füllfarbe auf eine RGB-Farbe, wobei jeder Farbwert einzeln angegeben wird. Die Werte für die Farben müssen zwischen 0 und 1 liegen.

```
void pdf_setrgbcolor_fill(int pdfhandle, double red, double green, double blue)
```

pdf_setrgbcolor_stroke

`pdf_setrgbcolor_stroke` setzt die Zeichenfarbe auf einen RGB-Wert, wobei jeder Farbwert einzeln angegeben wird. Die Werte für die Farben müssen zwischen 0 und 1 liegen. Die Funktion schließt außerdem die Zeichnung ab und stellt das Bild in der gewählten Farbe dar.

```
void pdf_setrgbcolor_stroke(int pdfhandle, double red, double green, double blue)
```

pdf_setrgbcolor

`pdf_setrgbcolor` setzt die Füll- und Zeichenfarbe auf einen RGB-Wert. Die Werte für die Farben müssen zwischen 0 und 1 liegen.

```
void pdf_setrgbcolor(int pdfhandle, double red, double green, double blue)
```

pdf_add_outline

`pdf_add_outline` setzt ein Lesezeichen auf die aktuelle Seite.

```
void pdf_add_outline(int pdfhandle, string bookmark)
```

pdf_set_transition

Die Funktion `pdf_set_transition` setzt einen Wert für den Übergang zwischen Seiten.

```
void pdf_set_transition(int pdfhandle, int transitiontype)
```

pdf_set_duration

`pdf_set_duration` setzt eine Verzögerung zwischen Seiten, die wirksam wird, wenn der Benutzer von Seite zu Seite springt. Die Angabe erfolgt in Sekunden, der Standardwert beträgt eine Sekunde.

```
void pdf_set_duration(int pdfhandle, double duration)
```

pdf_open_image_file

`pdf_open_image_file` öffnet ein Bild aus einer Bilddatei und gibt ein Handle darauf zurück.

```
int pdf_open_image_file(int pdfhandle, string type, string file)
```

pdf_open_memory_image

`pdf_open_memory_image` öffnet ein von PHP erzeugtes Bild und macht es für ein PDF-Dokument verfügbar.

```
int pdf_open_memory_image(int pdfhandle, string imagehandle)
```

pdf_close_image

`pdf_close_image` schließt ein Bild, das durch eine der `pdf_open_xxx`-Funktionen geöffnet wurde.

```
void pdf_close_image(int pdfhandle, int imagehandle)
```

pdf_place_image

pdf_place_image platziert ein Bild auf einer Seite.

```
void pdf_place_image(int pdfhandle, int imagehandle,  
                    double xpos, double ypos, double scale)
```

pdf_add_annotation

Diese Funktion fügt dem Dokument eine Notiz hinzu. Dabei kann die Größe und Lage des geöffneten Notizfensters auf der aktuellen Seite durch Festlegung der Ecken angegeben werden.

```
void pdf_add_annotation(int pdfhandle,  
                       double lowleftx, double lowlefty,  
                       double uprightright, double uprightright,  
                       string title, string annotation)
```

pdf_set_border_style

Diese Funktion legt den Rand um Hyperlinks und Notizen fest.

```
void pdf_set_border_style(int pdfhandle, string style, double width)
```

pdf_add_pdflink

Diese Funktion fügt einen Hyperlink in das Dokument ein. Angegeben wird die Position und das Sprungziel. Das Sprungziel muss ein PDF-Dokument sein, *page* bestimmt die Seite, auf die gesprungen wird.

```
void pdf_add_pdflink(int pdfhandle,  
                   double lowleftx, double lowlefty,  
                   double uprightright, double uprightright,  
                   string filename, int page, string dest)
```

pdf_add_weblink

Diese Funktion fügt einen Hyperlink auf eine externe Website hinzu.

```
void pdf_add_weblink(int pdfhandle,  
                   double lowleftx, double lowlefty,  
                   double uprightright, double uprightright, string url)
```

Shockwave-Flash-Funktionen

swf_openfile

Die Funktion öffnet eine neue Shockwave-Datei. Diese Funktion muss immer zuerst aufgerufen werden, da die anderen Funktionen nur in eine bereits existierende Datei schreiben können.

```
void swf_openfile(string filename, float width, float height,  
                  float framerate, float r, float g, float b)
```

swf_closefile

Diese Funktion schließt die aktuelle, zuvor geöffnete SWF-Datei.

```
void swf_closefile(void)
```

swf_labelframe

Das aktuelle Frame bekommt eine Bezeichnung.

```
void swf_labelframe(string name)
```

swf_showframe

Die Funktion gibt das aktuelle Frame aus.

```
void swf_showframe(void)
```

swf_setframe

Die Funktion wechselt zu dem mit `framenummer` ausgewählten Frame.

```
void swf_setframe(int framenummer)
```

swf_getframe

Die Funktion ermittelt die Nummer des aktuellen Frames.

```
int swf_getframe(void)
```

swf_mulcolor

Setzt die »global-multiply«-Farbe auf den angegebenen Farbwert.

```
void swf_mulcolor(float r, float g, float b, float a)
```

swf_addcolor

Setzt die »global-add«-Farbe auf den angegebenen Farbwert.

```
void swf_addcolor(float r, float g, float b, float a)
```

swf_placeobject

Die Funktion platziert ein Objekt auf dem Bildschirm.

```
void swf_placeobject(int objid, int depth)
```

swf_modifyobject

Diese Funktion modifiziert die aktuelle Position oder Farbe des aktuellen Objekts.

```
void swf_modifyobject(int depth, int how)
```

swf_removeobject

Diese Funktion entfernt das Objekt auf der Ebene *depth*.

```
void swf_removeobject(int depth)
```

swf_nextid

Die Funktion gibt die nächste frei verfügbare Objekt-ID zurück.

```
int swf_nextid(void)
```

swf_startdoaction

Die Funktion startet eine Beschreibung einer Aktion im aktuellen Frame. Sie müssen diese Funktion zuerst aufrufen, bevor irgendwelche Aktionen im Frame definiert werden.

```
void swf_startdoaction(void)
```

swf_enddoaction

Die Funktion beendet den aktuell laufenden Film im aktuellen Frame.

```
void swf_startdoaction(void)
```

swf_actiongotoframe

Mit dieser Funktion spielen Sie einen gespeicherten Film im ausgewählten Frame ab. Danach stoppt die Ausführung.

```
void swf_actiongotoframe(int framenummer)
```

swf_actiongeturl

Die Funktion ermittelt die URL aus einem Shockwave-Film mit der Adresse *url* und dem Ziel *target*.

```
void swf_actiongeturl(string url, string target)
```

swf_actionnextframe

Mit dieser Funktion wird ein Frame weiter geschaltet. Die Frames werden durch einen internen Zeiger adressiert.

```
void swf_actionnextframe(void)
```

swf_actionprevframe

Mit dieser Funktion wird ein Frame zurück geschaltet. Die Frames werden durch einen internen Zeiger adressiert.

```
void swf_actionnextframe(void)
```

swf_actionplay

Die Funktion spielt einen Film aus dem aktuellen Frame ab.

```
void swf_actionplay(void)
```

swf_actionstop

Die Funktion stoppt den Film aus dem aktuellen Frame.

```
void swf_actionstop(void)
```

swf_actiontogglequality

Mit dieser Funktion wird die Qualität des laufenden Films zwischen guter und schlechter Qualität umgeschaltet.

```
void swf_actiontogglequality(void)
```

swf_actionwaitforframe

Die Funktion überspringt Aktionen, wenn ein Frame nicht geladen wurde. Zuerst wird überprüft, ob das angegebene Frame geladen ist. Wenn das nicht der Fall ist, wird die Anzahl *skipcount* Aktionen übersprungen.

```
void swf_actionwaitforframe(int framenummer, int skipcount)
```

swf_actionsettarget

Mit dieser Funktion setzen Sie das Ziel für Aktionen. Damit können Sie mehrere Flash-Filme kontrollieren.

```
void swf_actionsettarget(string target)
```

swf_actiongotolabel

Diese Funktion zeigt einen mit der angegebenen Marke *label* markierten Frame an.

```
void swf_actiongotolabel(string label)
```

swf_defineline

Zeichnet eine Linie mit den angegebenen Parametern.

```
void swf_defineline(int objid, float x1, float y1, float x2, float y2, float width)
```

swf_definerect

Zeichnet ein Rechteck mit den angegebenen Parametern.

```
void swf_definerect(int objid, float x1, float y1, float x2, float y2, float width)
```

swf_startshape

Die Funktion startet eine komplexe Form im angegebenen Objekt.

```
void swf_startshape(int objid)
```

swf_definepoly

Die Funktion zeichnet ein Polygon, dessen Koordinaten in einem Array übergeben werden.

```
void swf_definepoly(int objid, array coords, int npoints, float width)
```

swf_shapelinesolid

Die Funktion bestimmt den Stil für Linien. Die nächste gezeichnete Linie erscheint mit diesen Attributen.

```
void swf_shapelinesolid(float r, float g, float b, float a, float width)
```

swf_shapefilloff

Schaltet das Ausfüllen für Formen aus. Nach Ausführung werden Formen nicht mehr gefüllt.

```
void swf_shapefilloff(void)
```

swf_shapefillsolid

Diese Funktion bestimmt, wie Formen gefüllt werden und verwendet dafür einen RGBA-Wert.

```
void swf_shapefillsolid(float r, float g, float b, float a)
```

swf_shapefillbitmapclip

Mit dieser Funktion wird ein Bitmap bestimmt, mit dem leere Flächen einer Form gefüllt werden.

```
void swf_shapefillbitmapclip(int bitmapid)
```

swf_shapefillbitmaptile

Mit dieser Funktion wird ein Bitmap bestimmt, mit dem leere Flächen einer Form im Kachelmode gefüllt werden.

```
void swf_shapefillbitmaptile(int bitmapid)
```

swf_shapemoveto

Diese Funktion verschiebt die Koordinaten des Zeichenwerkzeugs auf die angegebene Position.

```
void swf_shapemoveto(float x, float y)
```

swf_shapelineto

Zeichnet eine Linie von der aktuellen Position zu den angegebenen Koordinaten. Der Endpunkt wird dann zum aktuellen Punkt.

```
void swf_shapelineto(float x, float y)
```

swf_shapecurveto

Die Funktion zeichnet eine Bezier-Kurve von den aktuellen Koordinaten zu den angegebenen und setzt den aktuellen Zeiger auf den Endpunkt.

```
void swf_shapecurveto(float x1, float y1, float x2, float y2)
```

swf_shapecurveto3

Die Funktion zeichnet eine kubische Bezier-Kurve von den aktuellen Koordinaten zu den angegebenen und setzt den aktuellen Zeiger auf den Endpunkt.

```
void swf_shapecurveto3(float x1, float y1, float x2, float y2, float x3, float y3)
```

swf_shapearc

Die Funktion zeichnet einen Kreisbogen. Die Koordinaten *x* und *y* geben den Mittelpunkt des Kreises an.

```
void swf_shapearc(float x, float y, float r, float angl, float ang2)
```

swf_endshape

Die Funktion schließt die Definition der aktuell erstellten Form ab.

```
void swf_endshape(void)
```

swf_definefont

Die Funktion definiert für einen angegebenen Font eine ID und setzt ihn als aktuellen Font. Zukünftige Ausgaben werden den Font verwenden.

```
void swf_definefont(int fontid, string fontname)
```

swf_setfont

Die Funktion wählt einen zuvor mit `swf_definefont` definierten Font anhand der Font-ID aus.

```
void swf_setfont(int fontid)
```

swf_fontsize

Setzt die Größe für den aktuellen Font.

```
void swf_fontsize(float size)
```

swf_fontslant

Setzt die Schräge des aktuellen Fonts im angegebenen Winkel. Positive Werte stellen die Schrift nach rechts (kursiv), negative Werte nach links.

```
void swf_fontslant(float slant)
```

swf_fonttracking

Die Funktion bestimmt den Abstand der Buchstaben im Fließtext. Der aktuelle Wert wird nicht absolut gesetzt, sondern vergrößert (positiver Wert) oder verringert (negativer Wert).

```
void swf_fonttracking(float tracking)
```

swf_getfontinfo

Die Funktion gibt ein assoziatives Array mit Angaben über den aktuellen Font zurück.

```
array swf_getfontinfo(void)
```

swf_definetext

Schreibt einen Text in das aktuelle Objekt.

```
void swf_definetext(int objid, string str, int docenter)
```

swf_textwidth

Mit dieser Funktion wird die Breite des in *str* übergebenen Textes mit den aktuellen Font-Parametern berechnet. Der Text wird aber nicht geschrieben.

```
float swf_textwidth(string str)
```

swf_definebitmap

Die Funktion definiert ein Bitmap. Akzeptiert werden GIF, JPEG, RGB oder FI.

```
void swf_definebitmap(int objid, string image_name)
```

swf_getbitmapinfo

Die Funktion ermittelt die Attribute des angegebenen Bitmaps. Zurückgegeben wird ein assoziatives Array.

```
array swf_getbitmapinfo(int bitmapid)
```

swf_startsymbol

Die Funktion startet die Definition eines Symbols.

```
void swf_startsymbol(int objid)
```

swf_endsymbol

Mit dieser Funktion wird die Definition eines Symbols beendet.

```
void swf_endsymbol(void)
```

swf_startbutton

Startet die Definition einer Schaltfläche mit den angegebenen Parametern.

```
void swf_startbutton(int objid, int type)
```

swf_addbuttonrecord

Diese Funktion steuert die Eigenschaften (mögliche Zustände) der aktuellen Schaltfläche.

```
void swf_addbuttonrecord(int states, int shapeid, int depth)
```

swf_oncondition

Die Funktion definiert eine Aktion, die ausgelöst wird, wenn das spezifizierte Ereignis eintritt.

```
void swf_oncondition(int transition)
```

swf_endbutton

Schließt die Definition einer Schaltfläche ab.

```
void swf_endbutton(void)
```

swf_viewport

Mit dieser Funktion wird ein Bereich auf der Zeichenfläche definiert, in dem künftig Ausgaben erfolgen.

```
void swf_viewport(double xmin, double xmax, double ymin, double ymax)
```

swf_ortho

Die Funktion definiert eine orthografische Sicht auf den aktuellen Sichtbereich.

```
void swf_ortho(double xmin, double xmax, double ymin, double ymax,  
               double zmin, double zmax)
```

swf_ortho2

Die Funktion definiert eine 2D-Sicht auf den Sichtbereich.

```
void swf_ortho2(double xmin, double xmax, double ymin, double ymax)
```

swf_perspective

Transformiert Koordinaten perspektivisch.

```
void swf_perspective(double fovy, double aspect, double near, double far)
```

swf_polarview

Die Funktion definiert die Position des Betrachters mit Polarkoordinaten.

```
void swf_polarview(double dist, double azimuth, double incidence, double twist)
```

swf_lookat

Die Funktion bestimmt den Sichtwinkel zwischen zwei Betrachtungspunkten.

```
void swf_lookat(double view_x, double view_y, double view_z,  
                double ref_x, double ref_y, double ref_z, double twist)
```

swf_pushmatrix

Legt die aktuelle Transformationsmatrix auf dem Stapel ab.

```
void swf_pushmatrix(void)
```

swf_popmatrix

Holt die aktuelle Transformationsmatrix vom Stapel.

```
void swf_popmatrix(void)
```

swf_scale

Skaliert die aktuelle Transformation.

```
void swf_scale(double x, double y, double z)
```

swf_translate

Die Funktion übersetzt die aktuelle Transformation auf die angegebenen Koordinaten.

```
void swf_translate(double x, double y, double z)
```

swf_rotate

Die Funktion rotiert die aktuelle Transformation um den angegebenen Winkel *angle* und die Achse *axis*. Für *axis* können Sie "x", "y" oder "z" angeben.

```
void swf_rotate(double angle, string axis)
```

swf_posround

Die Funktion schaltet eine interne Rundungsfunktion ein oder aus. Wenn die Rundungsfunktion aktiviert ist, werden Texte unter Umständen exakter dargestellt, da berechnete Werte auf volle Pixel gesetzt werden.

```
void swf_posround(int round)
```

Mail-Funktionen (IMAP, POP3, NNTP)

mail

Die Funktion mail dient zum Versenden von E-Mail über einen SMTP-Server.

```
boolean mail(string to, string subject, string content [, string add_header])
```

imap_alerts

Die Funktion gibt alle Meldungen in einem Array zurück.

```
array imap_alerts(void)
```

imap_append

imap_append fügt eine Nachricht an eine Mailbox an.

```
int imap_append(int imapstream, string mailbox, string message, string flags)
```

imap_base64

imap_base64 dekodiert Base64-kodierten Text.

```
string imap_base64(string texttocode)
```

imap_binary

Die Funktion konvertiert einen 8-Bit-Text in das Base-64-Format.

```
string imap_binary(string binarystring)
```

imap_body

imap_body liest den Nachrichtentext einer Nachricht.

```
string imap_body(int imapstream, int messagenumber, int flags)
```

imap_check

imap_check prüft die aktuelle Mailbox.

```
object imap_check(int imapstream)
```

imap_clearflag_full

Die Funktion löscht ein bestimmtes Flag.

```
string imap_clearflag_full(int imapstream, string sequence,  
                           string flag, string options)
```

imap_close

imap_close schließt eine IMAP-Verbindung.

```
int imap_close(int imapstream [, int flags])
```

imap_createmailbox

imap_createmailbox erzeugt eine neue Mailbox mit dem Namen *mailbox*. Die Funktion gibt im Erfolgsfall TRUE zurück, sonst FALSE.

```
int imap_createmailbox(int imapstream, string mailbox)
```

imap_delete

imap_delete markiert eine Nachricht zum Löschen aus der Mailbox. Die Funktion gibt immer TRUE zurück. Die eigentliche Löschung der Nachrichten erfolgt mit der Funktion *imap_expunge*.

```
int imap_delete(int imapstream, int messagenumber)
```

imap_deletemailbox

imap_deletemailbox löscht eine Mailbox mit allen darin enthaltenen Nachrichten.

```
int imap_deletemailbox(int imapstream, string mailbox)
```

imap_errors

Die Funktion gibt alle Fehlermeldungen in einem Array zurück. Nach dem Ausführen der Funktion wird der Fehlerspeicher gelöscht.

```
array imap_errors(void)
```

imap_expunge

imap_expunge löscht alle Nachrichten, die zum Löschen gekennzeichnet wurden. Die Funktion gibt immer TRUE zurück.

```
int imap_expunge(int imapstream)
```

imap_fetchbody

imap_fetchbody liest einen bestimmten Teil einer Nachricht.

```
string imap_fetchbody(int imapstream, int messagenumber,  
                        string partnumber, int flags)
```

imap_fetchheader

Die Funktion holt Kopfzeilen unformatiert und unbehandelt und gibt sie als Zeichenketten zurück.

```
string imap_fetchheader(int imapstream, int message, int flags)
```

imap_fetchstructure

`imap_fetchstructure` liest die Struktur der Nachricht mit der Nummer *messagenumber*.

```
array imap_fetchstructure(int imapstream, int messagenumber)
```

imap_fetch_overview

Die Funktion liest die Kopfzeileninformationen aus und gibt sie in einem Array zurück.

```
array imap_fetch_overview(int imapstream, string sequence [, int flags])
```

imap_header

`imap_header` liest den Kopf (Header) einer Nachricht.

```
object imap_header(int imapstream, int messagenumber,  
                    int fromlength, int subjectlength, int defaulthost)
```

imap_headers

`imap_headers` gibt die Kopfzeilen aller Nachrichten der Mailbox zurück.

```
array imap_headers(int imapstream)
```

imap_last_error

Die Funktion gibt den Text des letzten Fehlers zurück, falls dieser aufgetreten ist.

```
string imap_last_error(void)
```

imap_listmailbox

`imap_listmailbox` liest alle Mailboxen aus.

```
array imap_listmailbox(int imapstream, string reference, string pattern)
```

imap_listsubscribed

`imap_listsubscribed` listet alle oder ausgewählte abonnierte Mailboxen auf.

```
array imap_listsubscribed(int imapstream, string reference, string pattern)
```

imap_mail

Die Funktion versendet eine E-Mail.

```
string imap_mail(string to, string subjectline, string content  
                [, string additionalheaders  
                [, string cc [, string bcc [, string returnpath]]]])
```

imap_mail_compose

Diese Funktion erzeugt eine MIME-Nachricht aus zwei Arrays, die Kopfzeilen und Inhaltszeilen enthalten.

```
string imap_mail_compose(array headers, array content)
```

imap_mail_copy

imap_mail_copy kopiert Nachrichten in die Mailbox.

```
int imap_mail_copy(int imapstream, string msglist, string mbox, int flags)
```

imap_mail_move

imap_mail_move verschiebt die Nachrichten *msglist* in die Mailbox *mbox*.

```
int imap_mail_move(int imapstream, string msglist, string mbox)
```

imap_num_msg

imap_num_msg ermittelt die Anzahl der Nachrichten in einer Mailbox.

```
int imap_num_msg(int imapstream)
```

imap_num_recent

imap_num_recent ermittelt die Anzahl der neuen Nachrichten in einer Mailbox.

```
int imap_num_recent(int imapstream)
```

imap_open

imap_open öffnet eine IMAP-Verbindung zu einer Mailbox.

```
int imap_open(string mailbox, string username, string password, int flags)
```

imap_ping

imap_ping prüft, ob die Verbindung noch besteht, und gibt dann TRUE zurück.

```
int imap_ping(int imapstream)
```

imap_renamemailbox

imap_renamemailbox benennt eine Mailbox in einen anderen Namen um. Der alte und der neue Name muss angegeben werden.

```
int imap_renamemailbox(int imapstream, string old_mailbox, string new_mailbox)
```

imap_reopen

imap_reopen eröffnet eine Verbindung erneut zu einer neuen Mailbox.

```
int imap_reopen(string imapstream, string mailbox [, string flags])
```

imap_subscribe

imap_subscribe abonniert eine Mailbox.

```
int imap_subscribe(int imapstream, string mailbox)
```

imap_undelete

imap_undelete nimmt ein Löschkennzeichen wieder weg.

```
int imap_undelete(int imapstream, int messagenumber)
```

imap_unsubscribe

Die Funktion imap_unsubscribe hebt das Abonnement der Mailbox *mailbox* wieder auf.

```
int imap_unsubscribe(int imapstream, string mailbox)
```

imap_qprint

imap_qprint konvertiert eine im Quoted-Printable-Format kodierte Zeichenkette in das 8-Bit-Format.

```
string imap_qprint(string stringtoconvert)
```

imap_8bit

imap_8bit konvertiert eine im 8-Bit-Format kodierte Zeichenkette in das Format Quoted-Printable.

```
string imap_8bit(string stringtoconvert)
```

imap_binary

imap_binary konvertiert eine im 8-Bit-Format kodierte Zeichenkette in das Base64-Format.

```
string imap_binary(string stringtoconvert)
```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

imap_scanmailbox

imap_scanmailbox liest eine Liste der Mailboxen ein und kann nach Mailboxnamen suchen.

```
array imap_scanmailbox(int imapstream, string searchbox)
```

imap_mailboxmsginfo

imap_mailboxmsginfo holt allgemeine Informationen über die Mailbox *imapstream*.

```
array imap_mailboxmsginfo(int imapstream)
```

imap_msgno

Diese Funktion ermittelt die Sequenznummer zu einer UID.

```
int imap_msgno(int imapstream, int uid)
```

imap_rfc822_write_address

imap_rfc822_write_address gibt eine nach RFC 822 formatierte Adresse zurück.

```
string imap_rfc822_write_address(string mailbox, string host, string personal)
```

imap_rfc822_parse_adrlist

imap_rfc822_parse_adrlist untersucht eine Adresse, in dem diese nach dem in RFC 822 definierten Verfahren analysiert wird.

```
string imap_rfc822_parse_adrlist(string address, string default_host)
```

imap_setflag_full

imap_setflag_full setzt Flags an Nachrichten.

```
string imap_setflag_full(int imapstream, string sequence, string flag, string opt)
```

imap_popen

imap_open öffnet eine persistente IMAP-Verbindung zu einer Mailbox.

```
int imap_popen(string mailbox, string username, string password, int flags)
```

imap_clearflag_full

imap_clearflag_full löscht Flags. Siehe auch Funktion imap_setflag_full.

```
string imap_clearflag_full(int imapstream, string sequence,  
                           string flag, string options)
```

imap_search

Mit Hilfe dieser Funktion kann in den Nachrichten auf dem IMAP-Server gesucht werden.

```
array imap_search(int imapstream, string searchword, int flags)
```

imap_status

Die Funktion gibt ein Status-Objekt zurück.

```
object imap_status(int imapstream, string mailbox, int options)
```

imap_sort

`imap_sort` gibt ein Array mit Nachrichtennummern zurück, die nach bestimmten Kriterien sortiert sind.

```
string imap_sort(int imapstream, int criteria, int reverse, int options)
```

imap_subscribe

Abonniert eine neue Mailbox. Dies trifft vor allem für Newsserver zu, wo mit dieser Funktion eine Newsgroup abonniert wird.

```
int imap_subscribe(int imapstream, string mailbox)
```

imap_fetchheader

`imap_fetchheader` gibt den Header einer Nachricht zurück.

```
string imap_fetchheader(int imapstream, int msgno, int flags)
```

imap_uid

`imap_uid` gibt die UID für eine Nachricht mit der Nummer *messagenumber* zurück.

```
string imap_uid(string mailbox, int messagenumber)
```

imap_undelete

Hebt die Löschung einer als gelöscht markierten Nachricht wieder auf.

```
int imap_undelete(int imapstream, int messagenumber)
```

imap_unsubscribe

Hebt das Abonnement einer Newsgroup wieder auf.

```
boolean imap_unsubscribe(int imapstream, string mailbox)
```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

imap_utf7_decode

Die Funktion wandelt einen 7-Bit-Text in das 8-Bit-Format um.

```
string imap_utf7_decode(string texttoconvert)
```

imap_utf7_encode

Die Funktion wandelt einen 8-Bit-Text in das 7-Bit-UTF-Format um.

```
string imap_utf7_encode(string texttoconvert)
```

imap_utf8

Die Funktion konvertiert Standardtext in das 8-Bit-UTF-Format.

```
string imap_utf8(string texttoconvert)
```

LDAP-Funktionen

ldap_add

ldap_add fügt einem LDAP-Verzeichnis Einträge hinzu.

```
int ldap_add(int linkhandle, string dn, array entry)
```

ldap_mod_add

ldap_mod_add fügt aktuellen Attributen weitere Werte hinzu.

```
int ldap_mod_add(int linkhandle, string dn, array entry)
```

ldap_mod_del

ldap_mod_del löscht Werte aktueller Attribute.

```
int ldap_mod_del(int linkhandle, string dn, array entry)
```

ldap_mod_replace

ldap_mod_replace ersetzt Werte von Attributen mit neuen Werten.

```
int ldap_mod_replace(int linkhandle, string dn, array entry)
```

ldap_bind

ldap_bind stellt die Bindung zu einem LDAP-Verzeichnis her.

```
int ldap_bind(int linkhandle [, string bind_rdn] [, string bind_password])
```

ldap_close

ldap_close schließt die Verbindung *linkhandle* zu einem LDAP-Server.

```
int ldap_close(int linkhandle)
```

ldap_connect

ldap_connect verbindet mit einem LDAP-Server.

```
int ldap_connect([string host] [,int portnumber])
```

ldap_count_entries

ldap_count_entries ermittelt die Anzahl der Einträge einer vorangegangenen Suche, deren Ergebnisliste durch *resulthandle* bezeichnet wird.

```
int ldap_count_entries(int linkhandle, int resulthandle)
```

ldap_delete

ldap_delete löscht ein durch *dn* bezeichnetes Objekt aus dem Verzeichnis.

```
int ldap_delete(int linkhandle, string dn)
```

ldap_dn2ufn

ldap_dn2ufn konvertiert ein DN *dn* in ein benutzerfreundlicheres Format, indem die Typbezeichner entfernt werden.

```
string ldap_dn2ufn(string dn)
```

ldap_explode_dn

ldap_explode_dn splittet ein durch *dn* bezeichnetes DN-Objekt in seine Bestandteile auf und gibt diese als assoziatives Array zurück.

```
array ldap_explode_dn(string dn, int with_attr)
```

ldap_first_attribute

ldap_first_attribute gibt das erste durch *resulthandle* bestimmte Attribut zurück.

```
string ldap_first_attribute(int linkhandle, int resulthandle, int identifier)
```

ldap_first_entry

ldap_first_entry gibt die erste ID der Ergebnisliste der Anfrage zurück.

```
int ldap_first_entry(int linkhandle, int resulthandle)
```

ldap_free_result

ldap_free_result gibt den von der Ergebnisliste belegten Speicher wieder frei.

```
int ldap_free_result(int resulthandle)
```

ldap_get_attributes

ldap_get_attributes holt die Attribute einer Suchanfrage.

```
array ldap_get_attributes(int linkhandle, int resulthandle)
```

ldap_get_dn

ldap_get_dn ermittelt den DN (distinguished name) des Eintrags.

```
string ldap_get_dn(int linkhandle, int resulthandle)
```

ldap_get_entries

ldap_get_entries gibt alle Objekte der Ergebnisliste zurück.

```
array ldap_get_entries(int linkhandle, int resulthandle)
```

ldap_get_values

ldap_get_values ruft alle Werte aus einer Ergebnisliste ab.

```
array ldap_get_values(int linkhandle, int resulthandle, string attribute)
```

ldap_list

ldap_list führt eine einfache Suche durch.

```
int ldap_list(int linkhandle, string base_dn, string filter [, array attributes])
```

ldap_modify

ldap_modify verändert einen LDAP-Eintrag.

```
int ldap_modify(int linkhandle, string dn, array entry)
```

ldap_next_attribute

ldap_next_attribute holt das nächste Attribut aus der Ergebnisliste. Dabei wird der interne Zeiger auf die Attribute weiter gesetzt.

```
string ldap_next_attribute(int linkhandle, int resulthandle, int identifier)
```

ldap_next_entry

ldap_next_entry holt den nächsten Eintrag.

```
int ldap_next_entry(int linkhandle, int resulthandle)
```

ldap_read

ldap_read sucht einen Eintrag.

```
int ldap_read(int linkhandle, string base_dn, string filter [, array attributes])
```

ldap_search

ldap_search sucht im LDAP-Baum.

```
int ldap_search(int linkhandle, string base_dn, string filter [, array attributes])
```

ldap_unbind

ldap_unbind hebt die Bindung an ein LDAP-Verzeichnis auf.

```
int ldap_unbind(int bindhandle)
```

Bildfunktionen (GD)

getimagesize

getimagesize gibt die Größe eines GIF-, JPG- oder PNG-Bildes zurück.

```
array getimagesize(string file [, array info])
```

imagearc

imagearc zeichnet einen Teil einer Ellipse.

```
int imagearc(int imgh, int cx, int cy, int w, int h, int s, int e, int col)
```

imagechar

imagechar zeichnet ein Zeichen in horizontaler Richtung.

```
int imagechar(int imgh, int font, int x, int y, string c, int col)
```

imagecharup

imagecharup zeichnet ein Zeichen in vertikaler Richtung.

```
int imagecharup(int imgh, int font, int x, int y, string c, int col)
```

imagecolorallocate

imagecolorallocate weist dem Bild eine Farbe zu. Der Farbwert wird zurückgegeben.

```
int imagecolorallocate(int imgh, int red, int green,int blue)
```

imagecolordeallocate

imagecolordeallocate hebt die Zuweisung einer Farbe wieder auf.

```
int imagecolorallocate(int imgh, int colorhandle)
```

imagecolortransparent

imagecolortransparent definiert eine Farbe als transparent. Der Farbwert muss zuvor mit imagecolorallocate bestimmt werden.

```
int imagecolortransparent(int imgh [, int color])
```

imagecopy

imagecopy kopiert von einem Bild in ein anderes.

```
int imagecopy(int imgh_dest, int imgh_source,  
              int dstX, int dstY, int srcX, int srcY, int srcW, int srcH )
```

imagecopymerge

imagecopymerge kopiert von einem Bild in ein anderes und vermischt die Bilddaten.

```
int imagecopygray(int imgh_dest, int imgh_source,  
                  int dstX, int dstY, int srcX, int srcY, int srcW, int srcH,  
                  int fact)
```

imagecopymergegray

imagecopymergegray kopiert von einem Bild in ein anderes und vermischt die Bilddaten auf Basis einer Grauskala.

```
int imagecopymergegray(int imgh_dest, int imgh_source,  
                       int dstX, int dstY, int srcX, int srcY, int srcW, int srcH,  
                       int fact)
```

imagecopyresized

imagecopyresized kopiert und skaliert einen Bildausschnitt von einem Bild in ein anderes.

```
int imagecopyresized(int imgh_dest, int imgh_source,  
                     int dstX, int dstY, int srcX, int srcY,  
                     int dstW, int dstH, int srcW, int srcH)
```

imagecreate

imagecreate erzeugt ein neues Bild und gibt ein Handle auf dieses Bild zurück.

```
int imagecreate(int x_size, int y_size)
```

imagecreatefromgif

imagecreatefromgif erzeugt ein neues Bild auf der Basis eines vorhandenen.

```
int imagecreatefromgif(string file)
```

imagecreatefromjpeg

imagecreatefromjpeg erzeugt ein neues Bild auf der Basis eines vorhandenen.

```
int imagecreatefromjpeg(string file)
```

imagecreatefrompng

imagecreatefrompng erzeugt ein neues PNG-Bild auf der Basis eines vorhandenen.

```
int imagecreatefrompng(string file)
```

imagecreatefromstring

imagecreatefromstring erzeugt ein neues Bild auf der Basis eines in einer binären Zeichenkette gegebenen.

```
int imagecreatefromstring(string imagedata)
```

imagecreatefromwbmp

imagecreatefromwbmp erzeugt ein neues WBMP-Bild auf der Basis eines vorhandenen. WBMP verwenden WAP-Handys.

```
int imagecreatefromwbmp(string file)
```

imagedashedline

imagedashedline zeichnet eine gestrichelte Linie.

```
int imagedashedline(int imgh, int x1, int y1, int x2, int y2, int col)
```

imagedestroy

imagedestroy zerstört ein im Speicher bearbeitetes Bild und gibt den Speicher frei.

```
int imagedestroy(int imgh)
```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

imagefill

`imagefill` füllt einen Bereich. Der Bereich muss von einer geschlossenen Linien begrenzt sein, sonst wird das gesamte Bild gefüllt.

```
int imagefill(int imgh, int x, int y, int col)
```

imagefilledpolygon

`imagefilledpolygon` zeichnet ein gefülltes Polygon (Vieleck) mit einer bestimmten Farbe.

```
int imagefilledpolygon(int imgh, array points, int num_points, int col)
```

imagefilledrectangle

`imagefilledrectangle` zeichnet ein gefülltes Rechteck in einer bestimmten Farbe.

```
int imagefilledrectangle(int imgh, int x1, int y1, int x2, int y2, int col)
```

imagefilltoborder

`imagefilltoborder` füllt einen Bereich mit einer bestimmten Farbe. Der Füllvorgang beginnt an der bestimmten Stelle *x, y*.

```
int imagefilltoborder(int imgh, int x, int y, int border, int col)
```

imagefontheight

`imagefontheight` ermittelt die Größe eines eingebauten Fonts in Pixel.

```
int imagefontheight(int font)
```

imagefontwidth

`imagefontwidth` ermittelt die Fontbreite in Pixel.

```
int imagefontwidth(int font)
```

imagegif

`imagegif` gibt die fertige Grafik im GIF-Format an den Browser oder in eine Datei aus.

```
int imagegif(int imgh [, string filename])
```

imagejpeg

`imagejpeg` gibt die fertige Grafik im JPEG-Format an den Browser oder in eine Datei aus.

```
int imagejpeg(int imgh [, string filename])
```

imagepng

imagepng gibt das fertige PNG-Bild an den Browser oder in eine Datei aus.

```
int imagepng(int imgh [, string filename])
```

imagewbmp

imagepng gibt das fertige WBMP-Bild an den Browser oder in eine Datei aus.

```
int imagewbmp(int imgh [, string filename])
```

imageinterlace

imageinterlace schaltet die Bildart interlaced oder nicht interlaced ein und aus. Im interlaced Modus werden Bilder nicht zeilenweise, sondern mit Versatz aufgebaut, was zu einer subjektiv schnelleren Anzeige führt. An der tatsächlichen Bildgröße ändert sich nichts.

```
int imageinterlace(int imgh [, int interlace])
```

imageline

imageline zeichnet eine Linie.

```
int imageline(int imgh, int x1, int y1, int x2, int y2, int col)
```

imageloadfont

imageloadfont lädt einen neuen Font.

```
int imageloadfont(string filename)
```

imagepolygon

imagepolygon zeichnet ein Polygon.

```
int imagepolygon(int imgh, array points, int numpoints, int col)
```

imagerectangle

imagerectangle zeichnet ein Rechteck.

```
int imagerectangle(int imgh, int x1, int y1, int x2, int y2, int col)
```

imagesetpixel

imagesetpixel setzt einen einzelnen Bildpunkt.

```
int imagesetpixel(int imgh, int x, int y, int col)
```

imagestring

imagestring zeichnet eine Zeichenkette horizontal.

```
int imagestring(int imgh, int font, int x, int y, string text, int col)
```

imagestringup

imagestringup zeichnet eine Zeichenkette horizontal.

```
int imagestringup(int imgh, int font, int x, int y, string text, int col)
```

imagesx

imagesx ermittelt die Breite eines Bildes in Pixel.

```
int imagesx(int imgh)
```

imagesy

imagesy ermittelt die Höhe eines Bildes in Pixel.

```
int imagesy(int imgh)
```

imaggettfbbox

imaggettfbbox gibt ein Array mit den Abmessungen eines Truetype-Textes zurück.

```
array imaggettfbbox(int size, int angle, string fontfile, string text)
```

imagetypes

imagetypes gibt ein Bitfeld mit der Codierung der Bildarten zurück, die die Bibliothek unterstützt.

```
int imagetypes(void)
```

imaggettfttext

imaggettfttext zeichnet einen Text in einem TrueType-Font.

```
array imaggettfttext(int imgh, int size, int angle,  
int x, int y, int col, string font, string text)
```

imagecolorat

imagecolorat ermittelt den Farbindex eines Bildpunktes.

```
int imagecolorat(int imgh, int x, int y)
```

imagecolorclosest

imagecolorclosest ermittelt den zu einem Farbwert passenden Index des in der aktuellen Farbpalette nächstliegenden Wertes.

```
int imagecolorclosest(int imgh, int red, int green, int blue)
```

imagecolorexact

imagecolorexact ermittelt den exakten Index der Farbe.

```
int imagecolorexact(int imgh, int red, int green, int blue)
```

imagecolorresolve

imagecolorresolve ermittelt den exakten Index der Farbe oder, wenn dies nicht erfolgreich ist, die nächstliegende Farbe.

```
int imagecolorresolve(int imgh, int red, int green, int blue)
```

imagecolorset

imagecolorset setzt eine Farbe auf einen Index der Palette.

```
bool imagecolorset(int imgh, int index, int red, int green, int blue)
```

imagecolorsforindex

imagecolorsforindex ermittelt die Farbe eines Indizes der aktuellen Palette des Bildes.

```
array imagecolorsforindex(int imgh, int index)
```

imagecolorstotal

imagecolorstotal ermittelt die Anzahl der Farben der aktuellen Palette des Bildes *imgh*.

```
int imagecolorstotal(int imgh)
```

imagepsloadfont

imagepsloadfont lädt einen Type-1-Font.

```
int imagepsloadfont(string filename)
```

imagepsfreefont

imagepsfreefont gibt den Speicher frei, der von einem Type-1-Font benutzt wird.

```
void imagepsfreefont(int pshdl)
```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

imagepsextextfont

imagepsextextfont erweitert oder verdichtet den Font.

```
void imagepsextextfont(int pshdl, double extends)
```

imagepsencodfont

imagepsencodfont ändert den Codevektor eines Fonts. Die Funktion lädt einen neuen Codevektor aus einer Datei, um Zeichen größer als 127 in einer Postscriptdatei zu verwenden.

```
int imagepsencodfont(string encodingfile)
```

imagepstext

imagepstext zeichnet eine Zeichenkette in einem Type-1-Font.

```
array imagepstext(int imgh, string text, int pshdl, int size,  
                  int foreground, int background,  
                  int x, int y, int space, int tightness,  
                  float angle, int antialias)
```

imagepsbbox

imagepsbbox ermittelt die Maße eines Textes, die in Anspruch genommen werden, wenn der Text erzeugt würde.

```
array imagepsbbox(string text, int font, int size,  
                  int space, int tightness, float angle)
```

imagepslantfont

Die Funktion stellt die Schriftneigung für einen geladenen Postscript-Font ein. Der Wert *angle* wird in Grad angegeben.

```
void imagepslantfont(int pshdl, double angle)
```

iptcparse

iptcparse zerlegt einen binären IPTC-Block (JPG-Bildinformationen) in einzelne Tags.

```
array iptcparse(string iptcblock)
```

Hyperwave-Funktionen

hw_array2objrec

Konvertiert die Daten aus einem Objekt-Array in einen Objekt-Datensatz, der als Zeichenkette zurückgegeben wird.

```
string hw_array2objrec(array object_array)
```

hw_children

Gibt ein Array mit Objekt-IDs zurück. Jede ID ist einem Teil des Kind *objectID* zugeordnet. Das Array enthält sowohl Dokumente als auch Kollektionen.

```
array hw_children(int connection, int objectID)
```

hw_childrenobj

Gibt ein Array mit Objekt-Arrays zurück. Jede ID ist einem Teil des Kind *objectID* zugeordnet. Das Array enthält sowohl Dokumente als auch Kollektionen.

```
array hw_childrenobj(int connection, int objectID)
```

hw_close

Schließt die Verbindung zu einem Hyperwave-Server. Die Funktion gibt FALSE zurück, wenn keine Verbindung bestand, sonst TRUE.

```
boolean hw_close(int connection)
```

hw_connect

hw_connect öffnet eine Verbindung zu einem Hyperwave-Server.

```
int hw_connect(string host, int port [, string username] [, string password])
```

hw_cp

hw_cp kopiert Objekte.

```
int hw_cp(int connection, array object_id_array, int destination_id)
```

hw_deleteobject

hw_deleteobject löscht Objekte.

```
int hw_deleteobject(int connection, int object_to_delete)
```

hw_docbyanchor

hw_docbyanchor gibt die Objekt-ID eines durch einen Anker adressierten Dokuments zurück.

```
int hw_docbyanchor(int connection, int anchorID)
```

hw_docbyanchorobj

hw_docbyanchorobj gibt ein Datensatzobjekt des Dokuments zurück, auf das *anchorID* zeigt.

```
string hw_docbyanchorobj(int connection, int anchorID)
```

hw_documentattributes

hw_documentattributes ermittelt das Datensatzobjekt eines Dokuments und gibt es als Zeichenkette zurück.

```
string hw_documentattributes(int hw_document)
```

hw_documentbodytag

hw_documentbodytag gibt das Body-Tag des Dokuments *hw_document* zurück.

```
string hw_documentbodytag(int hw_document)
```

hw_documentcontent

Die Funktionen hw_documentcontent gibt den Inhalt des Dokuments *hw_document* zurück.

```
string hw_documentcontent(int hw_document)
```

hw_documentsetcontent

hw_documentsetcontent setzt oder ersetzt den Inhalt eines Dokuments *hw_document*.

```
string hw_documentsetcontent(int hw_document, string content)
```

hw_documentsize

hw_documentsize ermittelt die Größe des Dokuments *hw_document* in Bytes.

```
int hw_documentsize(int hw_document)
```

hw_errormsg

hw_errormsg gibt die letzte Fehlermeldung zurück.

```
string hw_errormsg(int connection)
```

hw_edittext

hw_edittext lädt ein Text-Dokument auf den Hyperwave-Server.

```
int hw_edittext(int connection, int hw_document)
```

hw_error

hw_error gibt die letzte Fehlernummer zurück oder 0, wenn keine Fehler auftraten.

```
int hw_error(int connection)
```

hw_free_document

hw_free_document gibt den von einem Dokument belegten Speicher wieder frei.

```
int hw_free_document(int hw_document)
```

hw_getparents

hw_getparents gibt ein Array von übergeordneten Objekten zu der angegebenen Objekt-ID *objectID* zurück.

```
array hw_getparentsobj(int connection, int objectID)
```

hw_getparentsobj

hw_getparentsobj gibt ein Array von Datensätzen mit Informationen übergeordneter Objekte zurück.

```
array hw_getparentsobj(int connection, int objectID)
```

hw_getchildcoll

hw_getchildcoll gibt die Kollektion der untergeordneten Objekte in einem Array zurück.

```
array hw_getchildcoll(int connection, int objectID)
```

hw_getchildcollobj

hw_getchildcollobj ermittelt die Datensätze der untergeordneten Objekte.

```
array hw_getchildcollobj(int connection, int objectID)
```

hw_getremote

hw_getremote holt ein Dokument von einem anderen Server, üblicherweise über FTP, HTTP und einige Datenbanken.

```
int hw_getremote(int connection, int objectID)
```

hw_getremotechildren

hw_getremotechildren holt untergeordnete Komponenten eines Dokuments von einem anderen Server.

```
int hw_getremotechildren(int connection, string object_record)
```

hw_getsrcbydestobj

hw_getsrcbydestobj gibt einen Anker zurück, der auf ein Datensatzobjekt zeigt.

```
array hw_getsrcbydestobj(int connection, int objectID)
```

hw_getobject

hw_getobject gibt das Datensatzobjekt selbst zurück.

```
array hw_getobject(int connection, int objectID)
```

hw_getandlock

hw_getandlock gibt ein Datensatzobjekt zurück und verriegelt den Zugriff durch andere.

```
string hw_getandlock(int connection, int objectID)
```

hw_gettext

hw_gettext gibt das durch *objectID* bezeichnete Textdokument zurück.

```
int hw_gettext(int connection, int objectID, int rootID)
```

hw_getobjectbyquery

hw_getobjectbyquery gibt ein Objekt auf gefundene Dokumente nach einer Suchanfrage zurück.

```
array hw_getobjectbyquery(int connection, string query, int max_hits)
```

hw_getobjectbyqueryobj

hw_getobjectbyqueryobj gibt ein Array von Objekten auf gefundene Dokumente nach einer Suchanfrage zurück.

```
array hw_getobjectbyqueryobj(int connection, string query, int max_hits)
```

hw_getobjectbyquerycoll

hw_getobjectbyquerycoll gibt eine Kollektion von Dokumentenobjekten in einem Array zurück.

```
array hw_getobjectbyquerycoll(int connection, int objectID,  
                              string query, int max_hits)
```

hw_getobjectbyquerycollobj

hw_getobjectbyquerycollobj gibt eine Kollektion von Dokumentenobjekten in einem Array zurück. Die maximale Anzahl der Treffer wird mit *max_hits* angegeben, -1 steht für unbegrenzt.

```
array hw_getobjectbyquerycollobj(int connection, int objectID,  
                                 string query, int max_hits)
```

hw_getchilddoccoll

hw_getchilddoccoll gibt ein Array von Objekt-IDs auf untergeordnete Dokumente zurück.

```
array hw_getchilddoccoll(int connection, int objectID)
```

hw_getchilddoccollobj

hw_getchilddoccollobj gibt ein Array von Objekten auf untergeordnete Dokumente zurück.

```
array hw_getchilddoccollobj(int connection, int objectID)
```

hw_getanchors

hw_getanchors gibt die Objekt-IDs der Anker im Dokument zurück.

```
array hw_getanchors(int connection, int objectID)
```

hw_getanchorsobj

hw_getanchorsobj gibt die Datensatzobjekte der Anker im Dokument in einem Array zurück.

```
array hw_getanchorsobj(int connection, int objectID)
```

hw_mv

hw_mv verschiebt Objekte. Die Funktion gibt die Anzahl der verschobenen Objekte zurück.

```
int hw_mv(int connection, array object_id, int source_id, int destination_id)
```

hw_identify

hw_identify identifiziert sich als Nutzer mit den angegebenen Daten.

```
int hw_identify(string username, string password)
```

hw_incollections

hw_incollections prüft auf Objekt-IDs in einer Kollektion.

```
array hw_incollections(int connection, array object_id_array,  
                      array collection_id_array, int return_collections)
```

hw_info

hw_info ermittelt Informationen über die aktuelle Verbindung.

```
string hw_info(int connection)
```

hw_inscoll

hw_inscoll fügt eine Kollektion ein.

```
int hw_inscoll(int connection, int objectID, array object_array)
```

hw_insdoc

hw_insdoc fügt ein Dokument ein.

```
int hw_insdoc(int connection, int parentID, string object_record, string text)
```

hw_insertdocument

hw_insertdocument lädt ein Dokument auf den Hyperwave-Server *connection*.

```
int hw_insertdocument(int connection, int parentID, int hw_document)
```

hw_insertobject

hw_insertobject fügt ein Datensatzobjekt ein.

```
int hw_insertobject(int connection, string objectrec, string parameter)
```

hw_modifyobject

hw_modifyobject verändert ein Datensatzobjekt.

```
int hw_modifyobject(int connection, int object_to_change,  
                  array remove, array add, int mode)
```

hw_new_document

hw_new_document erzeugt ein neues Dokument im lokalen Speicher.

```
int hw_new_document(string object_record, string document_data, int document_size)
```

hw_objrec2array

hw_objrec2array konvertiert ein Datensatzobjekt in ein Array.

```
array hw_objrec2array(string objectrec)
```

hw_outputdocument

hw_outputdocument zeigt ein Dokument an.

```
int hw_outputdocument(int hw_document)
```

hw_pconnect

hw_pconnect stellt eine persistente (ständige) Verbindung mit einem Hyperwave-Server her. Die Funktion gibt ein Handle für die Verbindung zurück.

```
int hw_pconnect(string host, int port, string username, string password)
```

hw_pipedocument

hw_pipedocument lädt ein Dokument auf den Hyperwave-Server.

```
int hw_pipedocument(int connection, int objectID)
```

hw_root

hw_root gibt die ID des Root-Objekts zurück.

```
int hw_root(void)
```

hw_unlock

hw_unlock gibt ein Dokument wieder frei.

```
int hw_unlock(int connection, int objectID)
```

hw_who

hw_who gibt eine Liste der aktuell verbundenen Nutzer an.

```
array hw_who(int connection)
```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

hw_getusername

hw_getusername gibt den Namen des Nutzers der aktuellen Verbindung zurück.

```
string hw_getusername(int connection)
```

B.5 Datenbankfunktionen

DBA-Funktionen

dba_open

dba_open öffnet eine Text-Datenbank vom Typ *handler* und gibt ein Handle darauf zurück (Bei den anderen Funktion steht dafür *dbahandle*).

```
int dba_open(string file, string mode, string handler)
```

dba_popen

dba_popen öffnet eine Text-Datenbank vom Typ *handler* persistent und gibt ein Handle darauf zurück (Bei den anderen Funktion steht dafür *dbahandle*).

```
int dba_popen(string file, string mode, string handler)
```

dba_close

dba_close schließt eine Datenbankdatei.

```
bool dba_close(int dbahandle)
```

dba_exists

Die Funktion dba_exists ermittelt, ob zu einem Schlüssel ein Wert in der Datenbank existiert.

```
bool dba_exists(string key, int dbahandle)
```

dba_fetch

dba_fetch ermittelt einen Wert zu einem Schlüssel.

```
string dba_fetch(string key, int dbahandle)
```

dba_insert

dba_insert fügt den Wert zu einem Schlüssel ein.

```
int dba_insert(string key, string value, int dbahandle)
```

dba_replace

dba_replace ersetzt einen Wert.

```
bool dba_replace(string key, string value, int dbahandle)
```

dba_delete

dba_delete löscht einen Wert.

```
bool dba_delete(string key, int dbahandle)
```

dba_firstkey

dba_firstkey ermittelt den ersten Schlüssel in der Datenbank.

```
string dba_firstkey(int dbahandle)
```

dba_nextkey

dba_nextkey ermittelt den nächsten Schlüssel.

```
string dba_nextkey(int dbahandle)
```

dba_optimize

dba_optimize optimiert die Datendatei.

```
string dba_optimize(int dbahandle)
```

dba_sync

dba_sync schreibt die Daten physisch auf die Festplatte, da PHP die Verarbeitung zuerst im Speicher vornimmt.

```
string dba_sync(int dbahandle)
```

DBM-Funktionen

dbmopen

dbmopen öffnet eine dbm-Datenbank und gibt ein Handle darauf zurück.

```
int dbmopen(string filename, string flags)
```

dbmclose

dbmclose schließt eine Datenbankdatei.

```
bool dbmclose(int dbmhdl)
```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

dbmexists

Die Funktion `dbmexists` ermittelt, ob zu einem Schlüssel ein Wert in der Datenbank existiert.

```
bool dbmexists(int dbmhdl, string key)
```

dbmfetch

`dbmfetch` ermittelt einen Wert zu einem Schlüssel.

```
string dbmfetch(int dbmhdl, string key)
```

dbminsert

`dbminsert` fügt den Wert zu einem Schlüssel ein.

```
int dbminsert(int dbmhdl, string key, string value)
```

dbmreplace

`dbmreplace` ersetzt einen Wert.

```
bool dbmreplace(int dbmhdl, string key, string value)
```

dbmdelete

`dbmdelete` löscht einen Wert.

```
bool dbmdelete(int dbmhdl, string key)
```

dbmfirstkey

`dbmfirstkey` ermittelt den ersten Schlüssel in der Datenbank.

```
string dbmfirstkey(int dbmhdl)
```

dbmnextkey

`dbmnextkey` ermittelt den nächsten Schlüssel.

```
string dbmnextkey(int dbmhdl, string key)
```

dblist

`dblist` beschreibt die genutzte Bibliothek (meist »flat file«, also Textdateien).

```
string dblist(void)
```

dBase-Funktionen

dbase_create

Die Funktion `dbase_create` erzeugt eine neue dBase-Datenbank.

```
int dbase_create(string filename, array fields)
```

dbase_open

Die Funktion `dbase_open` öffnet eine bestehende dBase-Datenbank.

```
int dbase_open(string filename, int flags)
```

dbase_close

Die Funktion `dbase_close` schließt eine Datenbank wieder.

```
bool dbase_close(int dbmhd1)
```

dbase_pack

Die Funktion `dbase_pack` packt eine dBase-Datenbank. Unter Packen versteht dBase das endgültige Entfernen von zum Löschen markierten Datensätzen.

```
bool dbase_pack(int dbmhd1)
```

dbase_add_record

`dbase_add_record` fügt einer dBase-Datenbank einen neuen Datensatz hinzu.

```
bool dbase_add_record(int dbase_identifizier, array record)
```

dbase_replace_record

Die Funktion `dbase_replace_record` ersetzt einen Datensatz in einer dBase-Datenbank.

```
bool dbase_replace_record(int dbmhd1, array record, int record_number)
```

dbase_delete_record

Die Funktion `dbase_delete_record` löscht einen Datensatz aus der dBase-Datenbank.

```
bool dbase_delete_record(int dbmhd1, int record)
```

dbase_get_record

Die Funktion `dbase_get_record` holt die Daten eines Datensatzes aus der dBase-Datenbank.

```
array dbase_get_record(int dbmhdl, int record)
```

dbase_get_record_with_names

Die Funktion holt die Daten eines Datensatzes aus der dBase-Datenbank in ein assoziatives Array mit den Feldnamen als Schlüssel.

```
array dbase_get_record(int dbmhdl, int record)
```

dbase_numfields

Die Funktion `dbase_numfields` gibt die Anzahl der Felder in der Datenbank zurück.

```
int dbase_numfields(int dbmhdl)
```

dbase_numrecords

Die Funktion `dbase_numrecords` gibt die Anzahl der Datensätze in einer dBase-Datenbank zurück.

```
int dbase_numrecords(int dbmhdl)
```

filePro-Funktionen

filepro

Die Funktion `filepro` liest und prüft die Map-Datei einer FilePro-Datenbank.

```
boolean filepro(string directory)
```

filepro_fieldname

Die Funktion `filepro_fieldname` ermittelt den Namen eines Feldes.

```
string filepro_fieldname(int field_number)
```

filepro_fielddtype

`filepro_fielddtype` ermittelt den Datentyp eines Felds.

```
string filepro_fielddtype(int field_number)
```

filepro_fieldwidth

filepro_fieldwidth gibt die Breite eines Felds an.

```
int filepro_fieldwidth(int field_number)
```

filepro_retrieve

Die Funktion filepro_retrieve holt eine Spalte aus der Datenbank.

```
string filepro_retrieve(int row_number, int field_number)
```

filepro_fieldcount

filepro_fieldcount zählt die Anzahl der Felder (Spalten) der geöffneten filePro-Datenbank.

```
int filepro_fieldcount(void)
```

filepro_rowcount

filepro_rowcount gibt die Anzahl der Datensätze der geöffneten Datenbank zurück.

```
int filepro_rowcount(void)
```

Informix-Funktionen

ifx_connect

ifx_connect öffnet eine Informix-Server-Verbindung.

```
int ifx_connect([string database] [, string userid] [, string password])
```

ifx_pconnect

ifx_pconnect öffnet eine persistente Datenbankverbindung zu einem Informix-Server.

```
int ifx_pconnect([string database] [, string userid] [, string password])
```

ifx_close

ifx_close schließt eine Verbindung zu einer Informix-Datenbank.

```
int ifx_close([int linkhandle])
```

ifx_query

ifx_query sendet eine SQL-Abfrage an den Informix-Server.

```
int ifx_query(string query [, int linkhandle] [, int cursor_type] [, mixed blobid])
```

ifx_prepare

ifx_prepare bereitet eine SQL-Anweisung zur Ausführung vor.

```
int ifx_prepare(string query, int conn_id, int cursor_def, mixed blobidarray)
```

ifx_do

Die Funktion ifx_do führt eine vorbereitete SQL-Anweisung aus.

```
int ifx_do(int resulthandle)
```

ifx_error

ifx_error gibt den letzten Fehlercode zurück.

```
int ifx_error(void)
```

ifx_errormsg

ifx_errormsg gibt die letzte Fehlermeldung zurück.

```
string ifx_errormsg([int errorcode])
```

ifx_affected_rows

ifx_affected_rows gibt die Anzahl der bearbeiteten Datensätze der letzten SQL-Anweisung zurück.

```
int ifx_affected_rows(int resulthandle)
```

ifx_getsqlca

Die Funktion ifx_getsqlca holt den Inhalt aus *sqlca.sqlerrd[0..5]* nach einer Abfrage.

```
array ifx_getsqlca(int resulthandle)
```

ifx_fetch_row

ifx_fetch_row holt einen Datensatz als assoziatives Array.

```
array ifx_fetch_row(int resulthandle [, mixed position])
```

ifx_htmltbl_result

Die Funktion ifx_htmltbl_result formatiert eine Ergebnisliste als HTML-Tabelle.

```
int ifx_htmltbl_result(int resulthandle [, string html_options])
```

ifx_fieldtypes

ifx_fieldtypes listet alle Felder auf.

```
array ifx_fieldtypes(int resulthandle)
```

ifx_fieldproperties

ifx_fieldproperties gibt eine Liste mit den Feldeigenschaften zurück.

```
array ifx_fieldproperties(int resulthandle)
```

ifx_num_fields

ifx_num_fields gibt die Anzahl der Spalten einer Ergebnisliste zurück.

```
int ifx_num_fields(int resulthandle)
```

ifx_num_rows

ifx_num_rows gibt die Anzahl der Datensätze an, die in der Ergebnisliste stehen.

```
int ifx_num_rows(int resulthandle)
```

ifx_free_result

ifx_free_result gibt den von Ergebnislisten benötigten Speicher frei.

```
int ifx_free_result(int resulthandle)
```

ifx_create_char

ifx_create_char erzeugt ein Zeichen-Objekt.

```
int ifx_create_char(string param)
```

ifx_free_char

ifx_free_char löscht ein Zeichen-Objekt.

```
int ifx_free_char(int blobid)
```

ifx_update_char

ifx_update_char erneuert den Inhalt eines Zeichen-Objekts.

```
int ifx_update_char(int blobid, string content)
```

ifx_get_char

ifx_get_char gibt den Inhalt eines Zeichen-Objekts zurück.

```
int ifx_get_char(int blobid)
```

ifx_create_blob

ifx_create_blob erzeugt ein BLOB-Objekt.

```
int ifx_create_blob(int type, int mode, string param)
```

ifx_copy_blob

ifx_copy_blob verdoppelt das BLOB-Objekt.

```
int ifx_copy_blob(int blobid)
```

ifx_free_blob

ifx_free_blob löscht das BLOB-Objekt.

```
int ifx_free_blob(int blobid)
```

ifx_get_blob

ifx_get_blob gibt den Inhalt eines BLOB-Objekts zurück.

```
string int ifx_get_blob(int blobid)
```

ifx_update_blob

ifx_update_blob erneuert den Inhalt eines BLOB-Objekts.

```
void ifx_update_blob(int blobid, string content)
```

ifx_blobinfile_mode

ifx_blobinfile_mode setzt einen Standardmodus für alle Abfragen, die BLOB-Objekte beinhalten.

```
void ifx_blobinfile_mode(int mode)
```

ifx_textasvarchar

ifx_textasvarchar setzt den Standardmodus für BLOB-Objekte vom Typ »Text«.

```
void ifx_textasvarchar(int mode)
```

ifx_byteasvarchar

ifx_byteasvarchar setzt den Standardmodus für BLOB-Objekte vom Typ »Byte«.

```
void ifx_byteasvarchar(int mode)
```

ifx_nullformat

ifx_nullformat setzt den Standardwert für die Datensatzabfrage.

```
void ifx_nullformat(int mode)
```

ifxus_create_slob

ifxus_create_slob erzeugt ein SLOB-Objekt und öffnet es.

```
int ifxus_create_slob(int mode)
```

ifx_free_slob

ifx_free_slob gibt ein SLOB-Objekt frei.

```
int ifxus_free_slob(int blobid)
```

ifxus_close_slob

ifxus_close_slob löscht ein SLOB-Objekt.

```
int ifxus_close_slob(int blobid)
```

ifxus_open_slob

ifxus_open_slob öffnet ein SLOB-Objekt.

```
int ifxus_open_slob(long blobid, int mode)
```

ifxus_tell_slob

ifxus_tell_slob gibt die aktuelle Dateiposition zurück.

```
int ifxus_tell_slob(long blobid)
```

ifxus_seek_slob

ifxus_seek_slob setzt die Position des Zeigers in der Datei.

```
int ifxus_seek_blob(long blobid, int mode, long offset)
```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

ifxus_read_slob

ifxus_read_slob liest Bytes aus einem SLOB-Objekt.

```
int ifxus_read_slob(long blobid, long nbytes)
```

ifxus_write_slob

ifxus_write_slob schreibt eine Zeichenkette in ein SLOB-Objekt.

```
int ifxus_write_slob(long blobid, string content)
```

MS-SQL-Funktionen

mssql_affected_rows

Die Funktion mssql_affected_rows gibt die Anzahl der Datensätze zurück, die von der letzten Aktion betroffen waren.

```
int mssql_affected_rows([int linkhandle])
```

mssql_close

Schließt eine Datenbankverbindung.

```
int mssql_close([int linkhandle])
```

mssql_connect

mssql_connect öffnet eine Datenbankverbindung.

```
int mssql_connect(string server, string user, string password)
```

mssql_data_seek

Die Funktion mssql_data_seek bewegt den internen Datenbankcursor.

```
int mssql_data_seek(int resulthandle, int row_number)
```

mssql_fetch_array

Die Funktion mssql_fetch_array liest einen Datensatz komplett aus und gibt sie in einem Array zurück.

```
array mssql_fetch_array(int resulthandle)
```

mssql_fetch_field

mssql_fetch_field ermittelt Informationen zu Feldern.

```
object mssql_fetch_field(int resulthandle, int field_offset)
```

mssql_fetch_object

Die Funktion mssql_fetch_object holt einen Datensatz und legt ihn als Objekt ab.

```
int mssql_fetch_object(int resulthandle)
```

mssql_fetch_row

Die Funktion mssql_fetch_row holt einen Datensatz als indiziertes Array.

```
array mssql_fetch_row(int resulthandle)
```

mssql_field_seek

mssql_field_seek setzt den Feldoffset.

```
int mssql_field_seek(int resulthandle, int field_offset)
```

mssql_field_length

mssql_field_length ermittelt die Länge eines Feldes.

```
int mssql_field_length(int resulthandle [, int field_offset])
```

mssql_field_name

mssql_field_name ermittelt den Namen eines Felds anhand des Offsets.

```
string mssql_field_name(int resulthandle, int field_offset)
```

mssql_field_type

mssql_field_type ermittelt den Typ des Feldes.

```
string mssql_field_type(int resulthandle [, int field_offset])
```

mssql_free_result

mssql_free_result gibt den Speicher frei, der für die Ablage der Ergebnislisten verbraucht wurde.

```
int mssql_free_result(int resulthandle)
```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

mssql_num_fields

mssql_num_fields gibt die Anzahl der Felder in einer Ergebnisliste zurück.

```
int mssql_num_fields(int resulthandle)
```

mssql_num_rows

mssql_num_rows gibt die Anzahl der Datensätze der letzten Ergebnisliste zurück.

```
int mssql_num_rows(string resulthandle)
```

mssql_pconnect

Die Funktion mssql_pconnect öffnet eine persistente (ständige) Datenbankverbindung.

```
int mssql_pconnect(string server, string user, string password)
```

mssql_query

mssql_query sendet eine SQL-Abfrage an den MS SQL Server.

```
int mssql_query(string query, int linkhandle)
```

mssql_result

mssql_result holt ein Feld aus einer Ergebnisliste.

```
int mssql_result(int resulthandle, int i, mixed field)
```

mssql_select_db

Mit der Funktion mssql_select_db wird eine Datenbank ausgewählt.

```
int mssql_select_db(string database_name, int linkhandle)
```

mssql_get_last_message

mssql_get_last_message gibt die letzte Fehlermeldung des SQL-Servers zurück.

```
string mssql_get_last_message(void)
```

mssql_min_error_severity

mssql_min_error_severity setzt das Niveau, ab der Fehlernummern registriert werden.

```
void mssql_min_error_severity(int severitylevel)
```

mssql_min_message_severity

mssql_min_message_severity setzt das Niveau, ab der Fehlermeldungen registriert werden.

```
int mssql_min_message_severity(int severitylevel)
```

mSQL-Funktionen

mysql

Die Funktion mysql sendet eine SQL-Anweisung an die Datenbank.

```
int mysql(string database, string query, int linkhandle)
```

mysql_affected_rows

Die Funktion mysql_affected_rows gibt die Anzahl der von der letzten Anweisung behandelten Datensätze zurück.

```
int mysql_affected_rows(int query_identifizier)
```

mysql_close

Die Funktion mysql_close schließt eine mSQL-Verbindung.

```
int mysql_close(int linkhandle)
```

mysql_connect

Die Funktion mysql_connect öffnet eine mSQL-Verbindung.

```
int mysql_connect(string host)
```

mysql_create_db

Die Funktion mysql_create_db erzeugt eine Datenbank.

```
int mysql_create_db(string database [, int linkhandle])
```

mysql_data_seek

Die Funktion mysql_data_seek bewegt den internen Datensatzcursor.

```
int mysql_data_seek(int query_identifizier, int row_number)
```

mysql_dbname

Die Funktion `mysql_dbname` gibt den Namen der aktuellen Datenbank zurück.

```
string mysql_dbname(int query_identifizier, int i)
```

mysql_drop_db

Die Funktion `mysql_drop_db` löscht eine mSQL-Datenbank.

```
int mysql_drop_db(string database_name, int linkhandle)
```

mysql_error

Die Funktion `mysql_error` gibt die Fehlnachricht der letzten Aktion zurück.

```
string mysql_error(void)
```

mysql_fetch_array

`mysql_fetch_array` holt einen Datensatz in ein Array.

```
int mysql_fetch_array(int query_identifizier)
```

mysql_fetch_field

`mysql_fetch_field` liest ein Feld aus der Ergebnisliste aus.

```
object mysql_fetch_field(int query_identifizier, int offset)
```

mysql_fetch_object

`mysql_fetch_object` holt einen Datensatz der angegebenen Ergebnisliste in ein Objekt.

```
int mysql_fetch_object(int query_identifizier)
```

mysql_fetch_row

`mysql_fetch_row` liest einen Datensatz in ein indiziertes Array.

```
array mysql_fetch_row(int query_identifizier)
```

mysql_fieldname

`mysql_fieldname` gibt den Namen eines Felds zurück.

```
string mysql_fieldname(int query_identifizier, int field)
```

mysql_field_seek

mysql_field_seek setzt den Datensatzcursor.

```
int mysql_field_seek(int query_identifizier, int offset)
```

mysql_fieldtable

mysql_fieldtable ermittelt den Namen der verwendeten Tabelle zu einem Feld.

```
int mysql_fieldtable(int query_identifizier, int field)
```

mysql_fieldtype

mysql_fieldtype gibt den Feldtyp zurück.

```
string mysql_fieldtype(int query_identifizier, int i)
```

mysql_fieldflags

mysql_fieldflags gibt die Feldeigenschaften zurück.

```
string mysql_fieldflags(int query_identifizier, int i)
```

mysql_fieldlen

mysql_fieldlen gibt die Länge eines Felds zurück.

```
int mysql_fieldlen(int query_identifizier, int i)
```

mysql_free_result

Die Funktion mysql_free_result gibt den Speicher frei, der durch die aktuelle Ergebnisliste belegt wurde.

```
int mysql_free_result(int query_identifizier)
```

mysql_list_fields

mysql_list_fields gibt Informationen über Felder zurück.

```
int mysql_list_fields(string database, string table)
```

mysql_list_dbs

mysql_list_dbs zeigt auf alle Datenbanken.

```
int mysql_list_dbs(void)
```

mysql_list_tables

mysql_list_tables zeigt Tabelleninformationen an.

```
int mysql_list_tables(string database)
```

mysql_num_fields

mysql_num_fields gibt die Anzahl der Felder in der Ergebnisliste zurück.

```
int mysql_num_fields(int query_identifier)
```

mysql_num_rows

mysql_num_rows gibt die Anzahl der Reihen in einer Ergebnisliste zurück.

```
int mysql_num_rows(int query_identifier)
```

mysql_pconnect

Die Funktion öffnet eine persistente (ständige) Verbindung zu einer Datenbank. mysql_pconnect arbeitet ähnlich wie mysql_connect.

```
int mysql_pconnect(string host)
```

mysql_query

mysql_query sendet die SQL-Anweisung *query* an den Server.

```
int mysql_query(string query, int linkhandle)
```

mysql_regcase

mysql_regcase wandelt eine Zeichenkette so um, dass sie mit Hilfe eines regulären Ausdrucks ohne Rücksicht auf Groß- und Kleinschreibung getestet werden kann.

```
string mysql_regcase(string teststring)
```

mysql_result

mysql_result holt die Ergebnisse einer Abfrage in die Ergebnisliste.

```
int mysql_result(int query_identifier, int i, mixed field)
```

mysql_select_db

mysql_select_db wählt eine Datenbank aus.

```
int mysql_select_db(string databasee, int linkhandle)
```

mysql_tablename

mysql_tablename ermittelt den Namen einer Tabelle zu einem Feld.

```
string mysql_tablename(int query_identifizier, int field)
```

MySQL-Funktionen

mysql_affected_rows

mysql_affected_rows gibt die Anzahl der von der letzten Operation betroffenen Reihen zurück.

```
int mysql_affected_rows([int linkhandle])
```

mysql_change_user

Die Funktion meldet den Benutzer der aktuellen Verbindung *linkhandle* bei der Datenbank (!) neu an und setzt, wenn es angegeben wurde, die aktuelle Datenbank auf *database*.

```
int mysql_change_user(string user, string password  
[, string database [, int linkhandle]])
```

mysql_close

mysql_close schließt eine MySQL-Verbindung.

```
int mysql_close([int linkhandle])
```

mysql_connect

mysql_connect öffnet eine MySQL-Verbindung und gibt ein Handle auf die Verbindung zurück, dass in anderen Beispielen als *linkhandle* bezeichnet wird.

```
int mysql_connect([string host][:port] [, string user] [, string password])
```

mysql_create_db

mysql_create_db erzeugt eine neue Datenbank.

```
int mysql_create_db(string database [, int linkhandle])
```

mysql_data_seek

mysql_data_seek bewegt den Zeiger auf die Ergebnisliste.

```
int mysql_data_seek(int resulthandle, int row_number)
```

mysql_db_name

Die Funktion ermittelt die Namen von Datenbanken aus der Ergebnisliste *resulthandle* der Funktion `mysql_list_dbs`.

```
int mysql_db_name(int resulthandle, int row [, mixed field])
```

mysql_db_query

`mysql_db_query` sendet eine SQL-Abfrage an die ausgewählte Datenbank. Dabei wird die aktuelle Datenbank auf den hier angegebenen Wert *database* verändert.

```
int mysql_db_query(string database, string query [, int linkhandle])
```

mysql_drop_db

`mysql_drop_db` löscht eine Datenbank.

```
int mysql_drop_db(string database [, int linkhandle])
```

mysql_errno

`mysql_errno` gibt die letzte Fehlernummer aus.

```
int mysql_errno([int linkhandle])
```

mysql_error

`mysql_error` gibt den letzten Fehlertext aus.

```
string mysql_error([int linkhandle])
```

mysql_escape_string

`mysql_escape_string` maskiert für SQL kritischen Zeichen in einer Zeichenkette.

```
string mysql_escape_string(string query)
```

mysql_fetch_array

`mysql_fetch_array` holt einen Datensatz der Ergebnisliste in ein assoziatives Array.

```
array mysql_fetch_array(int resulthandle [, int result_typ])
```

mysql_fetch_assoc

`mysql_fetch_assoc` holt Informationen über eine Spalte der Ergebnisliste in ein assoziatives Array.

```
array mysql_fetch_assoc(int resulthandle)
```

mysql_fetch_field

mysql_fetch_field holt Informationen über eine Spalte der Ergebnisliste in ein Objekt.

```
object mysql_fetch_field(int resulthandle [, int field_offset])
```

mysql_fetch_lengths

mysql_fetch_lengths gibt den maximal benötigten Platz für die Ergebnisse in Form eines Arrays zurück.

```
array mysql_fetch_lengths(int resulthandle)
```

mysql_fetch_object

mysql_fetch_object holt einen Datensatz aus der Ergebnisliste in ein Objekt.

```
object mysql_fetch_object(int resulthandle [, int result_type])
```

mysql_fetch_row

mysql_fetch_row holt einen Datensatz aus der Ergebnisliste als ein numerisches Array.

```
array mysql_fetch_row(int resulthandle)
```

mysql_field_name

mysql_field_name ermittelt den Namen eines Feldes.

```
string mysql_field_name(int resulthandle, int index)
```

mysql_field_seek

mysql_field_seek setzt den Zeiger auf die Ergebnisliste.

```
int mysql_field_seek(int resulthandle, int offset)
```

mysql_field_table

mysql_field_table ermittelt den Namen einer Tabelle, in der sich das benannte Feld befindet.

```
string mysql_field_table(int resulthandle, int offset)
```

mysql_field_type

mysql_field_type ermittelt den Datentyp einer Spalte.

```
string mysql_field_type(int resulthandle, int offset)
```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

mysql_field_flags

mysql_field_flags ermittelt die Optionen eines Felds.

```
string mysql_field_flags(int resulthandle, int offset)
```

mysql_field_len

mysql_field_len ermittelt die Länge eines Felds.

```
int mysql_field_len(int resulthandle, int offset)
```

mysql_free_result

mysql_free_result gibt Speicher für Ergebnislisten wieder frei.

```
int mysql_free_result(int resulthandle)
```

mysql_insert_id

mysql_insert_id gibt die ID zurück, welche die letzte INSERT-Anweisung erzeugt hat.

```
int mysql_insert_id([int linkhandle])
```

mysql_list_fields

mysql_list_fields zeigt Daten der Felder der Tabelle an.

```
int mysql_list_fields(string database, string table [, int linkhandle])
```

mysql_list_dbs

mysql_list_dbs zeigt alle Datenbanken an.

```
int mysql_list_dbs([int linkhandle])
```

mysql_list_tables

mysql_list_tables zeigt alle Tabellen einer Datenbank an.

```
int mysql_list_tables(string database [, int linkhandle])
```

mysql_num_fields

mysql_num_fields ergibt die Anzahl der Felder der Ergebnisliste.

```
int mysql_num_fields(int resulthandle)
```

mysql_num_rows

mysql_num_rows Anzahl der Datensätze in der Ergebnisliste.

```
int mysql_num_rows(int resulthandle)
```

mysql_pconnect

mysql_pconnect öffnet eine persistente Verbindung.

```
int mysql_pconnect([string host] [: int port] [, string user] [, string password])
```

mysql_query

mysql_query sendet eine SQL-Abfrage an MySQL.

```
int mysql_query(string query [, int linkhandle])
```

mysql_result

mysql_result gibt den Inhalt genau eines Felds zurück.

```
int mysql_result(int resulthandle, int row, mixed field)
```

mysql_select_db

mysql_select_db wählt eine Datenbank als Standard aus.

```
int mysql_select_db(string database [, int linkhandle])
```

mysql_tablename

mysql_tablename ermittelt den Namen einer Tabelle zu einem Feld *ifield*.

```
string mysql_tablename(int resulthandle, int ifield)
```

ODBC-Funktionen

odbc_autocommit

odbc_autocommit schaltet die Behandlung der automatischen Bestätigung von Transaktionen um.

```
int odbc_autocommit(int connhandle [, int commit])
```

odbc_binmode

`odbc_binmode` behandelt Spalten mit binären Daten. Der Parameter *mode* bestimmt die betroffenen Datentypen (BINARY, VARBINARY, LONGVARBINARY).

```
int odbc_binmode(int resulthandle, int mode)
```

odbc_close

`odbc_close` schließt eine Verbindung zu einem über ODBC angeschlossenen Datenbankserver. Die Funktion schlägt fehl, wenn die Verbindung noch offene Transaktionen führt.

```
void odbc_close(int connhandle)
```

odbc_close_all

`odbc_close_all` schließt alle offenen Verbindungen.

```
void odbc_close_all(void)
```

odbc_commit

`odbc_commit` bestätigt eine Transaktion.

```
int odbc_commit(int connhandle)
```

odbc_connect

`odbc_connect` öffnet die Verbindung zu einer Datenquelle.

```
int odbc_connect(string dsn, string user, string password [, int cursor_type])
```

odbc_cursor

`odbc_cursor` ermittelt den Namen eines Cursors für eine Ergebnisliste *resulthandle*.

```
string odbc_cursor(int resulthandle)
```

odbc_exec

`odbc_exec` verarbeitet eine SQL-Anweisung und sendet sie zur Ausführung an den SQL-Server.

```
int odbc_exec(int connhandle, string query_string)
```

odbc_execute

`odbc_execute` führt eine vorbereitete Anweisung aus.

```
int odbc_execute(int resulthandle [, array parameters_array])
```

odbc_fetch_into

odbc_fetch_into holt einen Datensatz aus der Ergebnisliste in ein nummeriertes Array.

```
int odbc_fetch_into(int resulthandle, int rownumber, array result_array)
```

odbc_fetch_row

odbc_fetch_row holt einen Datensatz.

```
int odbc_fetch_row(int resulthandle [, int row_number])
```

odbc_field_name

odbc_field_name ermittelt den Spaltennamen für das Feld *field*.

```
string odbc_field_name(int resulthandle, int field)
```

odbc_field_type

odbc_field_type ermittelt den Datentyp für das Feld *field*.

```
string odbc_field_type(int resulthandle, int field)
```

odbc_field_len

odbc_field_len ermittelt die Länge in Bytes für das Feld *field*.

```
int odbc_field_len(int resulthandle, int field)
```

odbc_free_result

odbc_free_result gibt Speicher frei, der von der Ergebnisliste *resulthandle* benötigt wurde.

```
int odbc_free_result(int resulthandle)
```

odbc_longreadlen

odbc_longreadlen behandelt Spalten mit LONG-Datentypen (betrifft LONG, LONGVARBINARY).

```
int odbc_longreadlen(int resulthandle, int length)
```

odbc_num_fields

odbc_num_fields gibt die Anzahl der Spalten in einer Ergebnisliste *resulthandle* zurück.

```
int odbc_num_fields(int resulthandle)
```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

odbc_pconnect

odbc_pconnect öffnet eine persistente Verbindung.

```
int odbc_pconnect(string dsn, string user, string password [, int cursor_type])
```

odbc_prepare

odbc_prepare bereitet eine SQL-Anweisung zur Ausführung mit odbc_execute vor.

```
int odbc_prepare(int connhandle, string query_string)
```

odbc_num_rows

odbc_num_rows ergibt die Anzahl der Reihen in der Ergebnisliste *resulthandle*.

```
int odbc_num_rows(int resulthandle)
```

odbc_result

odbc_result gibt den Inhalt eines Felds der Ergebnisliste zurück. *field* kann die Nummer oder der Name eines Felds sein.

```
string odbc_result(int resulthandle, mixed field)
```

odbc_result_all

odbc_result_all gibt Ergebnisse als HTML-Tabelle zurück.

```
int odbc_result_all(int resulthandle [, string format])
```

odbc_rollback

odbc_rollback macht die innerhalb einer Transaktion vorgenommenen Änderungen an der Datenbank rückgängig.

```
int odbc_rollback(int connhandle)
```

odbc_setoption

Die Funktion odbc_setoption stellt verschiedene Parameter ein.

```
int odbc_setoption(int id, int function, int option, int param)
```

Oracle 8-Funktionen

ocidefinebyname

ocidefinebyname verwendet nutzerdefinierte PHP-Variablen, um Ergebnisse von SELECT-Abfragen auszuwerten.

```
int ocidefinebyname(int statement, string column, mixed &variable [, int type])
```

ocibindbyname

ocibindbyname bindet eine PHP-Variable an einen Platzhalter.

```
int ocibindbyname(int statement, string ph_name, mixed &var, int len [, int type])
```

ocilogon

ocilogon öffnet eine Verbindung zu Oracle.

```
int ocilogon(string user, string password [, string oracle_sid])
```

ociploton

ociploton öffnet eine persistente Verbindung zu Oracle.

```
int ociploton(string user, string password [, string oracle_sid])
```

ocinlogon

ocinlogon öffnet eine neue Verbindung zu Oracle.

```
int ocinlogon(string user, string password [, string oracle_sid])
```

ocilogoff

ocilogoff schließt die Verbindung *connhandle*.

```
int ocilogoff(int connhandle)
```

ociexecute

ociexecute führt eine vorbereitete Anweisung aus.

```
int ociexecute(int statement [, int mode])
```

ocicommit

ocicommit bestätigt offene, aber bereits abgearbeitete Transaktionen und macht die Daten in der Datenbank damit gültig.

```
int ocicommit(int connhandle)
```

ocirollback

ocirollback macht den von der letzten offenen Transaktion bewachten Vorgang rückgängig.

```
int ocirollback(int connhandle)
```

ocinewdescriptor

ocinewdescriptor initialisiert ein neues Handle auf ein leeres LOB bzw. eine Datei.

```
string ocinewdescriptor(int connhandle [, int descriptor_type])
```

ocirowcount

ocirowcount ergibt die Anzahl der betroffenen Datensätze. Diese Funktion kann nur mit UPDATE oder DELETE eingesetzt werden, nicht aber mit SELECT.

```
int ocirowcount(int statement)
```

ocinumcols

ocinumcols gibt die Anzahl der Spalten einer Abfrage zurück.

```
int ocinumcols(int statement)
```

ociresult

ociresult gibt den Wert eines Felds zurück.

```
mixed ociresult(int statement, mixed column_name)
```

ocifetch

ocifetch holt den nächsten Datensatz in einen internen Ergebnispufer.

```
mixed ocifetch(int statement)
```

ocifetchinto

ocifetchinto holt den nächsten Datensatz in ein Array.

```
int ocifetchinto(int statement, array &result [, int mode])
```

ocifetchstatement

ocifetchstatement holt den alle Datensätze in das Array *&result* und gibt die Anzahl zurück.

```
int ocifetchstatement(int statement, array &result)
```

ocicolumnisnull

ocicolumnisnull prüft, ob eine Ergebnisspalte der Anweisung *stmt* NULL ist.

```
int ocicolumnisnull(int statement, mixed column_name)
```

ocicolumnsize

Die Funktion ocicolumnsize gibt die Größe einer Ergebnisspalte zurück. Der Spaltenindex beginnt mit 1.

```
int ocicolumnsize(int statement, mixed column_name)
```

ocicolumnname

Die Funktion ocicolumnname gibt den Namen einer Spalte anhand der Nummer aus dem Ergebnissatz zurück.

```
int ocicolumnname(int statement, int column_number)
```

ocicolumnsize

Die Funktion ocicolumnsize gibt den Breite einer Spalte zurück, die als Nummer oder Name angegeben werden kann.

```
int ocicolumnsize(int statement, mixed column)
```

ocicolumntype

Die Funktion ocicolumntype gibt den Datentyp einer Spalte anhand der Nummer aus dem Ergebnissatz zurück.

```
int ocicolumntype(int statement, int column_number)
```

ociserverversion

Die Funktion ociserverversion gibt die Version des Oracle-Servers an.

```
int ociserverversion(int connhandle)
```

ocistatementtype

Die Funktion `ocistatementtype` gibt die Art der betreffenden Anweisung an (SELECT, INSERT, UPDATE usw.).

```
int ocistatementtype(int statement)
```

ocinewcursor

Die Funktion gibt einen neuen Zeiger auf einen Ergebnissatz zurück.

```
int ocinewcursor(int connhandle)
```

ocifreestatement

Die Funktion gibt alle von einem Ergebnissatz verbrauchten Ressourcen wieder frei.

```
int ocifreestatement(int statement)
```

ocifreecursor

Die Funktion gibt die Ressourcen eines Zeigers wieder frei.

```
int ocifreecursor(int statement)
```

ocifreedesc

Löscht ein Handle für ein großes (LOB)-Objekt.

```
bool ocifreedesc(object lob_object)
```

ociparse

Analysiert eine SQL-Anweisung und gibt ein Handle zurück. Dieses Handle wird in den anderen Funktionen als *statement* bezeichnet.

```
int ociparse(int connhandle, string query)
```

ocierror

Die Funktion gibt alle Fehler zurück, die mit dem Handle zusammenhängen.

```
array ocierror(int statement | int connhandle)
```

ociinternaldebug

Schaltet den internen Debugger der Oracle-Schnittstelle ein (Standard: aus, *switch* = 0).

```
int ociinternaldebug(int switch)
```

ocicancel

Die Funktion stoppt das Lesen von Daten vom Zeiger.

```
int ocicancel(int statement)
```

ocisetprefetch

Mit der Funktion wird festgelegt, wie viele Reihen (*numrows*) gelesen werden sollen.

```
int ocisetprefetch(int statement, int numrows)
```

ociwritelobtofile

Die Funktion schreibt Daten eines LOB-Objekts in eine binäre Datei.

```
int ociwritelobtofile(object lob [, string file [, int start [, int len]]])
```

ocisavelobfile

Die Funktion gibt den Inhalt eines LOB-Objekts als Zeichenkette zurück, sodass diese Daten gespeichert werden können.

```
string ocisavelobfile(object lob)
```

ocisavelob

Die Funktion gibt den Inhalt eines LOB-Objekts als Zeichenkette zurück.

```
string ocisavelob(object lob)
```

ociloadlob

Die Funktion lädt Daten in ein LOB-Objekt.

```
string ociloadlob(object lob)
```

ocicolumnscale

Ermittelt den Wertebereich (Anzahl der Stellen vor dem Komma) einer numerischen Spalte).

```
int ocicolumnscale(int statement, int column_number)
```

ocicolumnprecision

Ermittelt die Genauigkeit (Anzahl der Stellen nach dem Komma) einer numerischen Spalte.

```
int ocicolumnprecision(int statement, int column_number)
```

ocicolumntyperaw

Ermittelt den ursprünglichen Datentyp der Spalte (unabhängig von den darin befindlichen Daten).

```
int ocicolumntyperaw(int statement, int column_number)
```

PostgreSQL-Funktionen

pg_close

pg_close schließt eine Verbindung *connhandle*.

```
bool pg_close(int connhandle)
```

pg_cmdtuples

pg_cmdtuples gibt die Anzahl der von einer Operation betroffenen Datensätze zurück.

```
int pg_cmdtuples(int resulthandle)
```

pg_connect

pg_connect öffnet eine normale Verbindung zur Datenbank.

```
int pg_connect(string host, string port  
               [, string options] [, string tty], string dbname)
```

pg_dbname

pg_dbname gibt den Namen der Datenbank zurück, die von der angegebenen Verbindung *connhandle* angesprochen wird.

```
string pg_dbname(int connhandle)
```

pg_errormessage

pg_errormessage gibt Fehlermeldungen zurück.

```
string pg_errormessage(int connhandle)
```

pg_exec

pg_exec führt eine SQL-Abfrage aus.

```
int pg_exec(int connhandle, string query)
```

pg_fetch_array

`pg_fetch_array` liest einen Datensatz einer Ergebnisliste in ein assoziatives Array ein.

```
array pg_fetch_array(int resulthandle, int row)
```

pg_fetch_object

`pg_fetch_object` holt eine Reihe der Ergebnisliste als Objekt.

```
object pg_fetch_object(int resulthandle, int row)
```

pg_fetch_row

`pg_fetch_row` holt einen Datensatz der angegebenen Ergebnisliste und überträgt sie in ein indiziertes Array.

```
array pg_fetch_row(int resulthandle, int row)
```

pg_fieldisnull

`pg_fieldisnull` prüft, ob ein Feldinhalt NULL ist.

```
int pg_fieldisnull(int resulthandle, int row, mixed field)
```

pg_fieldname

`pg_fieldname` gibt einen Spaltennamen zurück. Die Spalten zählen von 0 beginnend in der Ergebnisliste *resulthandle*. Im Fehlerfall wird -1 zurückgegeben.

```
string pg_fieldname(int resulthandle, int field)
```

pg_fieldnum

`pg_fieldnum` gibt eine Spaltennummer zurück.

```
int pg_fieldnum(int resulthandle, string fieldname)
```

pg_fieldprtlen

`pg_fieldprtlen` gibt die Länge eines Felds beim Ausdruck des Inhalts zurück.

```
int pg_fieldprtlen(int resulthandle, int rownumber, string fieldname)
```

pg_fieldsize

`pg_fieldsize` gibt den internen Platzverbrauch eines Felds zurück.

```
int pg_fieldsize(int resulthandle, string fieldname)
```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

pg_fieldtype

pg_fieldtype gibt den Namen des Datentyps eines Felds zurück.

```
string pg_fieldtype(int resulthandle, int fieldnumber)
```

pg_freeresult

pg_freeresult gibt Speicher frei.

```
int pg_freeresult(int resulthandle)
```

pg_getlastoid

pg_getlastoid gibt die letzte *objid* zurück; dies ist die Adresse eines großen Objekts.

```
int pg_getlastoid(int resulthandle)
```

pg_host

pg_host gibt den Namen des PostgreSQL-Hosts zurück.

```
string pg_host(int connhandle)
```

pg_loclose

pg_loclose schließt ein großes Objekt, das durch *fd* adressiert wird.

```
void pg_loclose(int fd)
```

pg_locreate

pg_locreate erzeugt ein großes Objekt.

```
int pg_locreate(int connhandle)
```

pg_loopen

pg_loopen öffnet ein großes Objekt und gibt einen Zeiger darauf zurück, mit dem anderen Funktionen der Zugriff gewährt wird.

```
int pg_loopen(int connhandle, int objoid, string mode)
```

pg_loread

pg_loread liest ein großes Objekt.

```
string pg_loread(int fd, int length)
```

pg_loreadall

pg_loreadall liest ein großes Objekt komplett ein.

```
void pg_loreadall(int fd)
```

pg_lounlink

pg_lounlink löscht ein großes Objekt. *lobjid* identifiziert das große Objekt.

```
void pg_lounlink(int connhandle, int lobjid)
```

pg_lowrite

pg_lowrite schreibt ein großes Objekt.

```
int pg_lowrite(int fd, string buffer)
```

pg_numfields

pg_numfields gibt die Anzahl der Felder in einer Ergebnisliste zurück.

```
int pg_numfields(int resulthandle)
```

pg_numrows

pg_numrows gibt die Anzahl der von der letzten Abfrageoperation betroffenen Datensätze zurück.

```
int pg_numrows(int resulthandle)
```

pg_options

pg_options gibt die von der aktuellen oder angegebenen Verbindung genutzten Einstellungen als Zeichenkette zurück.

```
string pg_options(int connhandle)
```

pg_pconnect

pg_pconnect stellt eine persistente (ständige) Verbindung her.

```
int pg_pconnect(string host, string port, string options, string tty, string db)
```

pg_port

pg_port gibt die Portnummer der aktuellen Verbindung zum PostgreSQL-Server zurück.

```
int pg_port(int connhandle)
```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

pg_result

pg_result gibt Ergebnisse zurück.

```
mixed pg_result(int resulthandle, int rownumber, mixed fieldname)
```

pg_tty

pg_tty gibt den Namen des tty-Terminals zurück. Für das PostgreSQL-Verbindungshandle *connhandle* wird der serverseitige Debugger-Port zurückgegeben.

```
string pg_tty(int connhandle)
```

Sybase-Funktionen

sybase_affected_rows

sybase_affected_rows ergibt die Anzahl der bearbeiteten Datensätze der letzten SQL-Anweisung. sybase_affected_rows gibt die Anzahl der von INSERT, UPDATE oder DELETE behandelten Reihen zurück.

```
int sybase_affected_rows([int linkhandle])
```

sybase_close

sybase_close schließt eine Verbindung zu einer Sybase-Datenbank.

```
int sybase_close(int linkhandle)
```

sybase_connect

sybase_connect öffnet eine reguläre Verbindung zu einer Sybase-Datenbank.

```
int sybase_connect(string server, string user, string password)
```

sybase_data_seek

sybase_data_seek bewegt den internen Datensatzcursor.

```
int sybase_data_seek(int resulthandle, int rownumber)
```

sybase_fetch_array

Die Funktion sybase_fetch_array liest den aktuellen Datensatz aus der Ergebnisliste aus und übergibt ihn in einem Array.

```
int sybase_fetch_array(int resulthandle)
```

sybase_fetch_field

Die Funktion `sybase_fetch_field` liest den Wert eines bestimmten Felds anhand des Offsets (Feldnummer).

```
object sybase_fetch_field(int resulthandle, int offset)
```

sybase_fetch_object

Die Funktion `sybase_fetch_object` liest einen Datensatz in ein Objekt ein.

```
int sybase_fetch_object(int resulthandle)
```

sybase_fetch_row

`sybase_fetch_row` holt einen Datensatz in ein Array.

```
array sybase_fetch_row(int resulthandle)
```

sybase_field_seek

`sybase_field_seek` setzt den Datensatzzeiger auf eine bestimmte Position.

```
int sybase_field_seek(int resulthandle, int offset)
```

sybase_free_result

`sybase_free_result` gibt den Speicher frei, der durch die Ergebnisliste belegt wird.

```
int sybase_free_result(int resulthandle)
```

sybase_num_fields

`sybase_num_fields` gibt die Anzahl der Felder in einer Ergebnisliste zurück.

```
int sybase_num_fields(int resulthandle)
```

sybase_num_rows

`sybase_num_rows` gibt die Anzahl der Datensätze in der ausgewählten Ergebnisliste zurück.

```
int sybase_num_rows(string resulthandle)
```

sybase_pconnect

Die Funktion `sybase_pconnect` öffnet eine persistente (ständige) Verbindung.

```
int sybase_pconnect(string server, string user, string password)
```

sybase_query

sybase_query sendet eine SQL-Anweisung an den Sybase-Datenbankserver.

```
int sybase_query(string query, int linkhandle)
```

sybase_result

sybase_result holt die Ergebnisse und gibt ein Handle zurück.

```
int sybase_result(intresulthandle, int i, mixed field)
```

sybase_select_db

sybase_select_db wählt eine Datenbank aus.

```
int sybase_select_db(string database, int linkhandle)
```

B.6 Systemnahe Funktionen

Funktionen zur Ausgabesteuerung

flush

Die Funktion leert den Ausgabepuffer und sendet den Inhalt an den Webserver.

```
void flush(void)
```

ob_end_clean

Diese Funktion löscht den Pufferinhalt und schaltet die Pufferung anschließend aus.

```
void ob_end_clean(void)
```

ob_end_flush

Diese Funktion sendet den Pufferinhalt und schaltet die Pufferung anschließend aus.

```
void ob_end_flush(void)
```

ob_get_contents

Die Funktion liest den gesamten Inhalt des Puffers und gibt ihn als Zeichenkette zurück.

```
string ob_get_contents(void)
```

ob_get_length

Die Funktion gibt die Größe des Inhalts des Ausgabepuffers in Byte zurück.

```
int ob_get_length(void)
```

ob_implicit_flush

Die Funktion schaltet das implizite Leeren des Puffers ein. Nach jeder vollständigen Ausgabe, beispielsweise mit echo oder print, wird der Puffer geleert und der Inhalt gesendet.

```
void ob_implicit_flush([int flag])
```

ob_start

Die Funktion startet die Ausgabepufferung. Wird diese nicht gestartet, werden alle erzeugten Daten sofort zum Webserver gesendet.

```
void ob_start(void)
```

Verbindungsfunktionen

connection_aborted

connection_aborted gibt TRUE zurück, wenn der Client nicht mehr verbunden ist.

```
int connection_aborted(void)
```

connection_status

Die Funktion connection_status gibt einen Statusbericht in Form eines Bitfelds zurück.

```
int connection_status(void)
```

connection_timeout

connection_timeout gibt TRUE zurück, wenn das Skript wegen Zeitüberschreitung endet. Der Standardwert für die Zeitüberschreitung ist 30 Sekunden.

```
int connection_timeout(void)
```

ignore_user_abort

ignore_user_abort legt fest, ob ein Skript bei Zeitüberschreitung beendet werden soll.

```
int ignore_user_abort([int setting])
```

Datenverarbeitungsfunktionen

pack

pack packt Daten in eine binäre Form.

```
string pack(string format [, mixed args, ...])
```

serialize

serialize erzeugt ein speicherbares Format von Daten.

```
string serialize(mixed value)
```

unpack

unpack entpackt mit pack verpackte Daten in ein Array.

```
array unpack(string format, string data)
```

unserialize

unserialize erzeugt aus einer serialisierten Datensammlung Variablen.

```
mixed unserialize(string str)
```

Apache-Funktionen

apache_lookup_uri

apache_lookup_uri führt eine Teilabfrage einer spezifischen URI aus und gibt alle Informationen darüber zurück.

```
class apache_lookup_uri(string filename)
```

apache_note

apache_note setzt oder liest Apache-Nachrichten.

```
string apache_note(string note_name [, string note_value])
```

getallheaders

getallheaders holt alle HTTP-Header.

```
array getallheaders(void)
```

virtual

virtual erstellt eine Unterabfrage und schließt eine Datei ein.

```
int virtual(string filename)
```

Systemfunktionen

d1

d1 lädt eine PHP-Bibliothek zur Laufzeit (Windows und Unix).

```
int d1(string libraryname)
```

leak

leak verschwendet Speicher zu Testzwecken.

```
void leak(int bytes)
```

uniqid

uniqid erzeugt eine einmalige 32 Byte lange ID auf Basis des Präfix *prefix*.

```
int uniqid(string prefix)
```

Programmausführung

escapeshellcmd

escapeshellcmd markiert Metazeichen der Shell.

```
string escapeshellcmd(string command)
```

escapeshellarg

escapeshellarg markiert Metazeichen in Kommandozeilenargumenten.

```
string escapeshellarg(string arguments)
```

exec

exec führt ein externes Programm aus.

```
string exec(string command [, string array] [, int return_var])
```

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

system

system startet ein externes Kommando und zeigt die Ausgaben an.

```
string system(string command [, int returnvalues])
```

passthru

passthru führt ein externes Programm aus und zeigt die Ausgaben an.

```
string passthru(string command [, int return_var])
```

eval

eval wertet eine Zeichenkette als PHP-Code und führt den Code aus.

```
void eval(string code_str)
```

die

die gibt eine Meldung aus und beendet das Skript.

```
void die(string message)
```

exit

exit beendet das aktuelle Skript.

```
void exit(void)
```

define_syslog_variables

Die Funktion definiert die für openlog und syslog benötigten Variablen. Parameter sind nicht einstellbar.

```
void define_syslog_variables(void)
```

openlog

openlog öffnet eine Verbindung zum Systemprotokoll.

```
int openlog(string identification, int option, int facility)
```

syslog

syslog erzeugt eine Meldung für das Systemprotokoll.

```
int syslog(int priocode, string message)
```

closelog

closelog schließt die Verbindung zum Protokoll.

```
int closelog(void)
```

register_shutdown_function

register_shutdown_function registriert eine Funktion, die am Ende des Skripts automatisch aufgerufen wird.

```
int register_shutdown_function(string functionname)
```

sleep

sleep verzögert den Skriptablauf um eine Anzahl Sekunden.

```
void sleep(int seconds)
```

usleep

usleep verzögert den Ablauf um eine Anzahl Mikrosekunden.

```
void usleep(int microseconds)
```

Informationen über PHP und die Umgebung

getenv

getenv ermittelt den Wert einer Umgebungsvariablen.

```
string getenv(string varname)
```

get_cfg_var

get_cfg_var holt den Wert einer PHP-Konfigurationsoption.

```
string get_cfg_var(string varname)
```

get_current_user

get_current_user gibt den Namen des aktuellen Nutzers aus.

```
string get_current_user(void)
```

get_magic_quotes_gpc

get_magic_quotes_gpc gibt den Status der magic_quotes_gpc-Variablen zurück.

```
long get_magic_quotes_gpc(void)
```

get_magic_quotes_runtime

get_magic_quotes_runtime ermittelt den Status für magic_quotes zur Laufzeit.

```
long get_magic_quotes_runtime(void)
```

getlastmod

getlastmod zeigt den Zeitpunkt der letzten Änderung an.

```
int getlastmod(void)
```

getmyinode

getmyinode gibt den Inode des Skripts zurück.

```
int getmyinode(void)
```

getmypid

getmypid gibt die Prozess-ID zurück.

```
int getmypid(void)
```

getmyuid

getmyuid gibt die User-ID des Skripteigentümers zurück.

```
int getmyuid(void)
```

getrusage

getrusage ermittelt die Nutzung der aktuellen Ressourcen.

```
array getrusage([int who])
```

get_included_files

Diese Funktion gibt ein Array mit den Dateinamen zurück, die mit der Anweisung include_once eingebunden wurden.

```
array get_includes_files(void)
```

get_required_files

Diese Funktion gibt ein Array mit den Dateinamen zurück, die mit der Anweisung require_once eingebunden wurden.

```
array get_required_files(void)
```

highlight_file

Mit dieser Funktion wird der Inhalt einer Datei auf PHP- und HTML-Tags hin durchsucht und diese Information farblich abgesetzt als HTML-Code ausgegeben. Die Funktion sendet den Inhalt direkt an den Webserver.

```
void highlight_file(string scriptname)
```

highlight_string

Mit dieser Funktion wird der Inhalt einer Zeichenkette auf PHP- und HTML-Tags hin durchsucht und diese Information farblich abgesetzt als HTML-Code ausgegeben. Die Funktion sendet den Inhalt direkt an den Webserver.

```
void highlight_string(string codestring)
```

ini_get

`ini_get` holt den Wert einer Konfigurationsoption aus der Datei `PHP.INI`.

```
string ini_get(string varname)
```

ini_restore

`ini_restore` macht die letzte Änderung rückgängig und stellt den Originalwert wieder her.

```
string ini_restore(string varname)
```

ini_set

`ini_set` setzt den Wert einer Konfigurationsoption. Der alte Wert wird zurückgegeben oder `FALSE`, wenn die Option in `PHP.INI` nicht definiert war. Ein anderer Name für diese Funktion ist `ini_alter`.

```
string ini_set(string varname, string newvalue)
```

phpinfo

`phpinfo` gibt umfangreiche Informationen über die PHP-Installation in Form von Tabellen direkt zum Browser aus.

```
int phpinfo(void)
```

phpversion

`phpversion` zeigt die aktuelle Version an. Die Funktion gibt eine Zeichenkette mit der vollständigen Versionsinformation zurück.

```
string phpversion(void)
```

phpcredits

Die Funktion gibt eine Liste der Programmierer aus, die an PHP und der Entwicklung der Module mitgewirkt haben.

```
void phpcredits(void)
```

php_logo_guid

Die Funktion gibt einen Verweis auf das PHP 4-Logo zurück.

```
string php_logo_guid(void)
```

zend_logo_guid

Die Funktion gibt einen Verweis auf das PHP 4-Logo zurück.

```
string zend_logo_guid(void)
```

zend_version

Die Funktion gibt die Version der Zend-Engine an. PHP 4 verwendet Zend 1.0, PHP 5 basiert auf Zend 2.0.

```
string zend_version(void)
```

extension_loaded

`extension_loaded` ermittelt, ob eine Bibliothek oder Erweiterung verfügbar ist.

```
bool extension_loaded(string name)
```

php_sapi_name

Die Funktion gibt den Namen des Webserver-Interfaces zurück.

```
string php_sapi_name(void)
```

php_uname

Gibt den Namen des Betriebssystems zurück.

```
string php_uname(void)
```

putenv

`putenv` setzt den Wert einer Umgebungsvariablen.

```
void putenv(string setting)
```

set_magic_quotes_runtime

set_magic_quotes_runtime setzt den aktuellen Wert der Variablen magic_quotes_runtime.

```
long set_magic_quotes_runtime(int new_setting)
```

set_time_limit

set_time_limit begrenzt die Laufzeit eines Skripts.

```
void set_time_limit(int seconds)
```

var_dump

Die Funktion gibt eine Variable aus, einschließlich einer Darstellung des Datentyps und des Inhalts.

```
void var_dump(mixed expression)
```

print_r

Die Funktion gibt ähnlich wie var_dump Informationen über eine Variable aus. Die Darstellung ist umfangreicher, aber auch unübersichtlicher.

```
void print_r(mixed expression)
```

get_browser

Ermittelt Informationen über den aktuell vom Benutzer verwendeten Browser. Zurückgegeben wird ein Objekt, das mit (array) in ein assoziatives Array gewandelt werden kann. Die Schlüssel entsprechen den Zeilen der entsprechenden Einträge in der Datei BROWSCAP.INI.

```
object get_browser(string useragent)
```

get_defined_functions

Diese Funktion gibt ein Array zurück, das alle benutzerdefinierten und systemeigenen Funktionsnamen enthält.

```
array get_defined_functions(void)
```

get_defined_vars

Die Funktion gibt alle globalen Variablen in einem Array zurück. Die Variablennamen bilden Schlüssel, die Werte enthalten die Inhalte der Variablen.

```
array get_defined_vars(void)
```

get_extension_funcs

Diese Funktion gibt ein Array mit den Namen der Funktionen einer Erweiterung zurück.

```
array get_extension_funcs(string extensionname)
```

get_loaded_extensions

Die Funktion gibt ein Array mit den Namen der geladenen Erweiterungen zurück.

```
array get_loaded_extensions(void)
```

get_resource_type

Diese Funktion ermittelt den Typ einer Ressource anhand des übergebenen Handles.

```
string get_resource_type(int handle)
```

register_tick_function

Registriert eine Funktion, die bei jedem Verarbeiten einer Zeile bzw. einem Vielfachen davon durch den Parser aufgerufen wird. Damit ist eine pseudoparallele Verarbeitung möglich.

```
string register_tick_function(string function_name [, mixed arguments])
```

unregister_tick_function

Hebt die Registrierung einer Tick-Funktion (`register_tick_function`) wieder auf.

```
void unregister_tick_function(string function_name)
```

Debugging, Fehlersuche, Protokollierung

assert

Untersucht eine PHP-Anweisung und gibt bei Erfolg TRUE zurück. Dies kann zum Testen von Skriptblöcken verwendet werden.

```
int assert(string string_assertion | boolean boolean_assertion)
```

assert_options

Steuert verschiedene Optionen für assert.

```
mixed assert_options(int option, mixed value)
```

debugger_on

debugger_on startet den Debugger.

```
int debugger_on(string address)
```

debugger_off

debugger_off stoppt den Debugger.

```
int debugger_off(void)
```

error_log

error_log gibt eine Fehlermeldung aus.

```
int error_log(string message, int message_type,  
              , string destination) [, string extra_headers])
```

error_reporting

error_reporting legt fest, welche Fehler angezeigt werden.

```
int error_reporting([int level])
```

function_exists

function_exists gibt TRUE zurück, wenn die benannte Funktion existiert.

```
int function_exists(string function_name)
```

set_error_handler

set_error_handler setzt, welche Behandlungsroutinen für Fehler verwendet werden.

```
string set_error_handler(string error_handler)
```

restore_error_handler

restore_error_handler stellt die ursprünglichen Fehlerbehandlungsroutinen wieder her.

```
int restore_error_handler(void)
```

trigger_error

trigger_error löst einen benutzerdefinierten Fehler aus und übergibt eine Meldung sowie das Fehlerniveau der Fehlerbehandlungsroutine.

```
void trigger_error(string errormessage, int errorlevel)
```


B.7 MySQL-Referenz

Auf den folgenden Seiten finden Sie eine Kurzreferenz zu den Befehlen und Funktionen, die MySQL verarbeiten kann.

Datentypen

Datentyp	Beschreibung
TINYINT [(M)] [UNSIGNED] [Z]	Kleine Ganzzahlen, von 0 bis 255 oder -128 bis 127
SMALLINT [(M)] [UNSIGNED] [Z]	Ganzzahlen, entweder von 0 bis 65 535 oder von -32 768 bis +32 767
MEDIUMINT [(M)] [UNSIGNED] [Z]	Ganzzahlen, entweder von 0 bis 16 777 215 oder von -8 388 608 bis +8 388 607
INT [(M)] [UNSIGNED] [Z] INTEGER	Ganzzahlen, entweder von 0 bis 4 294 967 295 oder von -2 147 283 648 bis +2 147 283 647
BIGINT [(M)] [UNSIGNED] [Z]	Ganzzahlen, entweder von 0 bis 18 446 744 073 709 551 615 oder von -9 223 372 036 854 775 808 bis +9 223 372 036 854 775 807
FLOAT(<i>precision</i>) [Z]	Fließkommazahl, immer vorzeichenbehaftet. <i>precision</i> kann 4 oder 8 sein; 4 steht für einfache Genauigkeit, 8 für doppelte.
FLOAT[(M,D)] [Z]	Der Wertebereich reicht von -3,402823466 ³⁸ bis -1,175494351 ⁻³⁸ , die 0 und +1,175494351 ⁻³⁸ bis +3,402823466 ³⁸
DOUBLE[(M,D)] [Z] DOUBLEPRECISION[(M,D)] [Z] REAL[(M,D)] [Z]	Wertebereich von -1,7976931348623157 ³⁰⁸ bis -2,2250738585072014 ⁻³⁰⁸ , die 0 und -2,2250738585072014 ⁻³⁰⁸ bis +1,7976931348623157 ³⁰⁸
DECIMAL(M,D) [Z] NUMERIC	Ungepackte Fließkommazahl, immer vorzeichenbehaftet. Zahlen werden als Zeichenkette gespeichert, jede Ziffer steht in einem Byte. Das Komma, Vorzeichen usw. belegen jeweils ein Byte.
DATE	Datum im Format "YYYY-MM-DD". Der Wertebereich geht von 1.1.1000 bis zum 31.12.9999.
DATETIME	Datum und Zeit im Format "YYYY-MM-DD hh:mm:ss".

Datentyp	Beschreibung
TIMESTAMP[(M)]	Zeitstempel, Wertebereich von 1.1.1970 bis zum 31.12.2036. Für die Angabe des Anzeigebereiches <i>M</i> gilt: • 14: YYYYMMDDhhmmss • 12: YYMMDDhhmmss • 8: YYYYMMDD • 6: YYMMDD
TIME	Zeit im Wertebereich von -838:59:59 bis +838:59:59 mit dem Ausgabeformat "hh:mm:ss"
YEAR	Jahr, Wertebereich 1901 bis 2155, Ausgabe YYYY.
CHAR(M) [BINARY]	Zeichenkette fester Länge <i>M</i> . Wertebereich 1 bis 255. Leerzeichen am Ende werden automatisch für die Ausgabe entfernt. Sortieren und Selektieren berücksichtigt Groß- und Kleinschreibung nicht, wenn Sie nicht BINARY verwenden.
VARCHAR(M) [BINARY]	Zeichenkette variabler Länge, maximal <i>M</i> . Wertebereich 1 bis 255. Leerzeichen am Ende werden automatisch für die Ausgabe entfernt. Sortieren und Selektieren berücksichtigt Groß- und Kleinschreibung nicht, wenn Sie nicht BINARY verwenden.
TINYBLOB, TINYTEXT	BLOB oder TEXT mit maximal 255 Byte
BLOB, TEXT	BLOB oder TEXT mit maximal 65 535 Byte
MEDIUMBLOB, MEDIUMTEXT	BLOB oder TEXT mit maximal 16 777 215 Byte
LONGBLOB, LONGTEXT	BLOB oder TEXT mit maximal 4 294 967 295 Byte
ENUM('wert1', 'wert2', ...,)	Aufzählung. Ein Feld dieses Typs kann nur eine Zeichenkette enthalten, die einem Objekt der Aufzählung entspricht.
SET('wert1', 'wert2', ...,)	Wie ENUM, kann aber mehrere Werte aus der Liste enthalten.

Mathematische Funktionen

Funktion	Beschreibung
ABS(x)	Absoluter Betrag der Zahl x
ACOS(x)	Arcuskosinus
ASIN(num)	Arcussinus
ATAN(num)	Arcustangens

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Funktion	Beschreibung
ATN2(<i>num1</i> , <i>num2</i>)	Arcustangens (Winkel) zwischen <i>num1</i> und <i>num2</i>
CEILING(<i>x</i>)	Die kleinste Ganzzahl größer oder gleich dem Ausdruck
COS(<i>x</i>)	Kosinus
COT(<i>x</i>)	Kotangens
DEGREES(<i>x</i>)	Umrechnung Radiant in Grad
EXP(<i>x</i>)	Potenz zur Basis e
FLOOR(<i>x</i>)	Die größte Ganzzahl kleiner oder gleich dem angegebenen numerischen Ausdruck.
GREATEST(<i>x1</i> , <i>x2</i> ,...)	Gibt den größten Wert der Liste zurück.
LEAST(<i>x1</i> , <i>x2</i> ,...)	Gibt den kleinsten Wert der Liste zurück.
LOG(<i>x</i>)	Natürlicher Logarithmus
LOG10(<i>x</i>)	Dekadischer Logarithmus
MOD(<i>n</i> , <i>m</i>)	Modulus, Rest einer Ganzzahldivision
PI()	Konstante π
POWER(<i>x</i> , <i>y</i>)	Potenz
RADIANS(<i>x</i>)	Umrechnung in Radiant
RAND(<i>x</i>)	Zufallszahl, <i>x</i> ist optional
ROUND(<i>x</i> , <i>d</i>)	Rundet Werte mathematisch, Stellen <i>d</i> sind optional.
SIGN(<i>x</i>)	Vorzeichen (gibt -1, 0 oder 1 zurück)
SIN(<i>x</i>)	Sinus
SQRT(<i>x</i>)	Quadratwurzel
TAN(<i>x</i>)	Tangens
TRUNCATE(<i>x</i> , <i>d</i>)	Gibt die Zahl <i>x</i> zurück, gekürzt auf <i>d</i> Dezimalstellen.

Logische Funktionen

Funktion	Beschreibung
ISNULL(<i>expr</i>)	Wertet den Ausdruck <i>expr</i> aus und gibt 1 zurück, wenn der Ausdruck NULL ist, sonst 0.
IFNULL(<i>expr1</i> , <i>expr2</i>)	Wertet den Ausdruck <i>expr1</i> aus und gibt <i>expr2</i> zurück, wenn der Ausdruck NULL ist, sonst <i>expr1</i> .
IF(<i>expr1</i> , <i>expr2</i> , <i>expr3</i>)	Wenn der Ausdruck <i>expr1</i> Wahr ist, wird <i>expr2</i> zurückgegeben, sonst <i>expr3</i> .

Funktionen für spaltenweise Berechnungen

Funktion	Beschreibung
AVG	<i>Average</i> . Der Durchschnitt der Felder.
COUNT	<i>Count</i> . Die Anzahl der Felder.
COUNT (*)	Repräsentiert die Anzahl aller Datensätze einer Tabelle.
SUM	<i>Summary</i> . Die Summe der Felder (Addition).
MAX	<i>Maximum</i> . Das Feld mit dem größten Wert bestimmt das Ergebnis.
MIN	<i>Minimum</i> . Das Feld mit dem kleinsten Wert bestimmt das Ergebnis.
STD STDDEV	Statistische <i>Standardabweichung</i> aller Werte der Liste.
BIT_OR	Bitweises <i>Oder</i>
BIT_AND	Bitweises <i>Und</i>

Funktionen für Systeminformationen

Funktion	Beschreibung
DATABASE()	Gibt den Namen der aktuellen Datenbank aus.
USER() SYSTEM_USER() SESSION_USER()	Aktueller MySQL-Nutzername
PASSWORD(<i>str</i>)	Erzeugt ein Kennwort zur Zeichenkette <i>str</i> .
ENCRYPT(<i>str</i> , <i>seed</i>)	Erzeugt ein Kennwort zur Zeichenkette <i>str</i> und mit dem Startwert <i>seed</i> . Nutzt das Unix-Kommando crypt. Unter Windows wird NULL zurückgegeben.
LAST_INSERT_ID()	Gibt den zuletzt erzeugten Wert einer AUTO_INCREMENT-Spalte zurück.
FORMAT(<i>n</i> , <i>d</i>)	Formatiert eine Zahl <i>n</i> mit Kommas als Tausendergruppensymbol und Punkt als Dezimaltrennzeichen mit <i>d</i> Dezimalstellen.
VERSION()	Gibt die Versionsnummer des MySQL-Server an.
GET_LOCK(<i>str</i> , <i>to</i>)	Erzeugt eine Verriegelung (Lock) mit dem Namen <i>str</i> und dem Zeitüberschreitungswert <i>to</i> .
RELEASE_LOCK(<i>str</i>)	Gibt die Verriegelung <i>str</i> wieder frei.

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Zeichenkettenfunktionen

Funktion	Beschreibung
ASCII(<i>str</i>)	Gibt den ASCII-Code des Zeichens <i>str</i> zurück. Hat <i>str</i> mehr als ein Zeichen, wird nur das erste Zeichen überprüft.
CHAR(<i>n</i> ,...)	Wandelt die Zahlen <i>n</i> in die entsprechenden ASCII-Zeichen um. Mehrere Argumente werden zu einer Zeichenkette kombiniert.
CONCAT(<i>str</i> ,...)	Verknüpft alle Argumente zu einer Zeichenkette. Wenn eines der Argumente NULL ist, wird NULL zurückgegeben.
LENGTH(<i>str</i>)	Länge der Zeichenketten <i>str</i> .
LOCATE(<i>sub</i> , <i>str</i>) POSITION(<i>sub</i> IN <i>str</i>)	Bestimmt die Position der Zeichenkette <i>sub</i> in der Zeichenkette <i>str</i> .
INSTR(<i>str</i> , <i>sub</i>)	Entspricht LOCATE, nur die Argumente sind vertauscht.
LPAD(<i>str</i> , <i>len</i> , <i>pad</i>)	Fügt <i>pad</i> links an <i>str</i> an, gibt jedoch nur <i>len</i> Zeichen zurück.
RPAD(<i>str</i> , <i>len</i> , <i>pad</i>)	Fügt <i>pad</i> rechts an <i>str</i> an, gibt jedoch nur <i>len</i> Zeichen zurück.
LEFT(<i>str</i> , <i>len</i>)	Gibt <i>len</i> Zeichen vom linken Ende der Zeichenkette <i>str</i> zurück.
RIGHT(<i>str</i> , <i>len</i>)	Gibt <i>len</i> Zeichen vom rechten Ende der Zeichenkette <i>str</i> zurück.
MID(<i>str</i> , <i>pos</i> , <i>len</i>)	Gibt <i>len</i> Zeichen von Position <i>pos</i> an der Zeichenkette <i>str</i> zurück.
SUBSTRING(<i>str</i> FROM <i>len</i> SUBSTRING(<i>str</i> , <i>pos</i> , <i>len</i>) SUBSTRING(<i>str</i> FROM <i>pos</i> FOR <i>len</i>)	Andere Schreibweisen für RIGHT und MID.
SUBSTRING(<i>str</i> , <i>pos</i>)	Gibt Teile von <i>str</i> ab Position <i>pos</i> zurück.
SUBSTRING_INDEX(<i>str</i> , <i>delimiter</i> , <i>count</i>)	Gibt den linken Teil einer Zeichenkette zurück, nachdem <i>count</i> mal das Zeichen <i>delimiter</i> aufgetreten ist.
LTRIM(<i>str</i>)	Entfernt Leerzeichen vom linken Ende.
RTRIM(<i>str</i>)	Entfernt Leerzeichen vom rechten Ende.
TRIM(<i>str</i>)	Entfernt Leerzeichen von beiden Enden der Zeichenkette <i>str</i> .
TRIM BOTH LEADING TRAILING <i>rem</i> FROM <i>str</i>	BOTH entspricht TRIM, LEADING entspricht LTRIM, TRAILING entspricht RTRIM, FROM ist optional, <i>rem</i> ist optional und steht für das zu entfernende Zeichen, <i>str</i> wird bearbeitet.
SOUNDEX(<i>str</i>)	Gibt die Lautfolge für <i>str</i> zurück.
SPACE(<i>n</i>)	Gibt <i>n</i> Leerzeichen zurück.

Funktion	Beschreibung
REPLACE(<i>str</i> , <i>from</i> , <i>to</i>)	Ersetzt alle Vorkommen von <i>from</i> in der Zeichenkette <i>str</i> durch <i>to</i> .
REPEAT(<i>str</i> , <i>count</i>)	Wiederholt die Zeichenkette <i>str</i> <i>count</i> mal.
REVERSE(<i>str</i>)	Dreht eine Zeichenkette um.
INSERT(<i>str</i> , <i>st</i> , <i>len</i> , <i>new</i>)	Fügt <i>len</i> Zeichen der Zeichenkette <i>new</i> an der Stelle <i>st</i> der Zeichenkette <i>str</i> ein.
ELT(<i>n</i> , <i>str1</i> , <i>str2</i> ,...)	Gibt die durch <i>n</i> bezeichnete Zeichenkette zurück: <i>str1</i> , wenn <i>n</i> =1 usw.
FIELD(<i>str</i> , <i>str1</i> , <i>str2</i> ..)	Gibt die Position von <i>str</i> in <i>str1</i> , <i>str2</i> usw. zurück: Wenn <i>str2</i> = <i>str</i> , wird 2 zurückgegeben.
FIND_IN_SET(<i>str</i> , <i>list</i>)	Gibt die Position von <i>str</i> in der Liste <i>list</i> zurück. Die Liste besteht aus kommaseparieren Werten.
MAKE_SET(<i>bits</i> , <i>list</i>)	Wählt die Elemente der Liste anhand der gesetzten Bits in <i>bits</i> aus.
LCASE, LOWER(<i>str</i>)	Wandelt in Kleinbuchstaben um
UCASE, UPPER(<i>str</i>)	Wandelt in Großbuchstaben um

Zahlenformatierungen

Funktion	Beschreibung
CONV(<i>n</i> , <i>from</i> , <i>to</i>)	Konvertiert Zahlen zwischen verschiedenen Zahlenbasen. Zurückgegeben wird immer eine Zeichenkette mit der ermittelten Zahl. <i>n</i> ist die Zahl, <i>from</i> die ursprüngliche Zahlenbasis, <i>to</i> die Zielbasis.
BIN(<i>n</i>)	Gibt eine Zahl <i>n</i> als Zeichenkette im Binärformat zurück, BIN(7) → "111".
OCT(<i>n</i>)	Gibt eine Zahl <i>n</i> als Zeichenkette im Oktalformat zurück.
HEX(<i>n</i>)	Gibt eine Zahl <i>n</i> als Zeichenkette im Hexadezimalformat zurück.

Funktionen zur Behandlung von Datums- und Zeitwerten

Allgemeine Datums- und Zeitfunktionen

Funktion	Beschreibung
DAYOFWEEK(<i>date</i>)	Tag der Woche, 1 ist Sonntag (1 – 7)
WEEKDAY(<i>date</i>)	Tag der Woche, 0 ist Montag (0 – 6)

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Funktion	Beschreibung
DAYOFMONTH(<i>date</i>)	Tag des Monats (1 – 31)
DAYOFYEAR(<i>date</i>)	Tag des Jahres (1 – 366)
MONTH(<i>date</i>)	Monat (1 – 12)
DAYNAME(<i>date</i>)	Wochentag (englisch, ausgeschrieben)
MONTHNAME(<i>date</i>)	Monat (englisch, ausgeschrieben)
QUARTER(<i>date</i>)	Quartal (1 – 4)
WEEK(<i>date</i>) WEEK(<i>date</i> , <i>first</i>)	Woche im Jahr (0 – 52). Das optionale Argument <i>first</i> bestimmt, welcher Wochentag als Beginn gezählt wird. 0=Sonntag.
YEAR(<i>date</i>)	Jahr (1000 – 9999)
HOURL(<i>time</i>)	Stunde (0 – 23)
MINUTE(<i>time</i>)	Minute (0 – 59)
SECOND(<i>time</i>)	Sekunde (0 – 59)
PERIOD_ADD(<i>p</i> , <i>n</i>)	Addiert <i>n</i> Monate zur Periode <i>p</i> . Die Periode wird im Format YYYYMM oder YYMM erwartet. Zurückgegeben wird immer die Langform YYYYMM.
PERIOD_DIFF(<i>p1</i> , <i>p2</i>)	Gibt die Differenz in Monaten zwischen <i>p1</i> und <i>p2</i> zurück.
TO_DAYS(<i>date</i>)	Gibt die Anzahl der Tage seit dem Jahr 0 zurück.
FROM_DAYS(<i>dn</i>)	Ermittelt ein Datum aus der Tageszahl <i>dn</i> .
CURDATE() CURRENT_DATE	Das aktuelle Datum (Systemzeit des Servers)
CURTIME() CURRENT_TIME	Die aktuelle Zeit (Systemzeit des Servers)
NOW() SYSDATE() CURRENT_TIMESTAMP	Datum und Uhrzeit (Systemzeit des Servers)
UNIX_TIMESTAMP	Unix Timestamp (Sekunden in GMT seit den 1.1.1970, 000 Uhr). Die Funktion wird auch von der Windows-Version unterstützt.
FROM_UNIXTIME(<i>stp</i>)	Gibt ein Datum entsprechend dem Unix Timestamp <i>stp</i> zurück.
FROM_UNIXTIME(<i>stp</i> , <i>format</i>)	Gibt ein Datum entsprechend dem Unix Timestamp <i>stp</i> zurück. Das Datum ist entsprechend <i>format</i> formatiert. Formatangaben entnehmen Sie der Tabelle <i>Platzhalter für Datumsformatierungen</i> weiter unten.

Funktion	Beschreibung
<code>SEC_TO_TIME(<i>sec</i>)</code>	Rechnet die Angabe in Sekunden in das Format HH:MM:SS um.
<code>TIME_TO_SEC(<i>time</i>)</code>	Rechnet eine Zeitangabe in Sekunden um.

Zeittypen für Datumsberechnungen

Typ-Code	Bedeutung	Format für <i>expr</i>
SECOND	Sekunde	ss
MINUTE	Minute	mm
HOURL	Stunde	hh
DAY	Tag	DD
MONTH	Monat	MM
YEAR	Jahr	YY
MINUTE_SECONDS	Minute und Sekunde	"mm:ss"
HOURL_MINUTE	Stunde und Minute	"hh:mm"
DAY_HOUR	Tage und Stunde	"DD hh"
YEAR_MONTH	Jahr und Monat	"YY-MM"
HOURL_SECOND	Stunde, Minute, Sekunde	"hh:mm:ss"
DAY_MINUTE	Tag, Stunde, Minute	"DD hh:mm"
DAY_SECONDS	Tag, Stunde, Minute, Sekunde	"DD hh:mm:ss"

Funktionen zur Datumsformatierung

Funktion	Berechnung
<code>DATE_FORMAT(<i>date</i>, <i>format</i>)</code>	Formatiert den Wert <i>date</i> mit dem Format <i>format</i> , dessen Elemente in der folgenden Tabelle <i>Platzhalter für Datumsformatierungen</i> beschrieben werden.
<code>TIME_FORMAT(<i>time</i>, <i>format</i>)</code>	Formatiert den Wert <i>time</i> mit dem Format <i>format</i> , dessen Elemente in der folgenden Tabelle <i>Platzhalter für Datumsformatierungen</i> beschrieben werden.

Platzhalter für Datumsformatierungen

Code	Bedeutung (Wertebereich)	Code	Bedeutung (Wertebereich)
%M	Monatsname (January – December)	%k	Stunde (0 – 23) ohne führende Null

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Code	Bedeutung (Wertebereich)	Code	Bedeutung (Wertebereich)
%W	Wochenname (Monday – Sunday)	%h	Stunde (01 – 12) mit führender Null
%D	Monat mit engl. Suffix (1 st , 2 nd , 3 rd usw.)	%I	Stunde (1 – 12) ohne führende Null
%Y	Jahr mit 4 Stellen	%l	Minuten (0 – 59)
%y	Jahr mit 2 Stellen	%i	Minuten (00 – 59) mit führender Null
%a	Abgekürzter Wochentag (Mon – Sun)	%n	Zeit, 12-Stunden-Format: hh:mm:ss AM PM
%d	Tag des Monats (00 – 31)	%T	Zeit, 24-Stunden-Format: hh:mm:ss
%e	Tag des Monats (0 – 31)	%S	Sekunde (00 – 59) mit führender Null
%m	Monat (00 – 12) mit führender Null	%s	Sekunde (0 – 59)
%c	Monat (0 – 12) ohne führende Null	%p	AM oder PM
%b	Abgekürzter Monatsname (Jan–Dec)	%w	Wochentag (0=Sonntag, 6=Samstag)
%j	Tag des Jahres (000 – 366)	%U	Woche, Sonntag ist der erste Tag der Woche (00 – 52)
%H	Stunde (00 – 23) mit führender Null	%u	Woche, Montag ist der erste Tag der Woche (00 – 52)
%%	Das Prozentzeichen selbst		

MySQL-Anweisungen

Anweisung	Beschreibung
CREATE DATABASE [IF NOT EXISTS] <i>db</i>	Erzeugt eine neue Datenbank <i>db</i>
DROP DATABASE [IF EXISTS] <i>db</i>	Löscht eine Datenbank <i>db</i>
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] <i>tbl</i> [(<i>create_def</i> ,...)] [<i>table_options</i>] [<i>select_statement</i>]	Erzeugt eine neue Tabelle <i>tbl</i> <i>create_def</i> kann folgendes sein: <ul style="list-style-type: none"> • <i>col_name type</i> [NOT NULL NULL] [DEFAULT <i>default_value</i>] [AUTO INCREMENT] [PRIMARY KEY] [<i>reference_def</i>] • PRIMARY KEY (<i>index_col</i>,...) • KEY [<i>index_name</i>] (<i>index_col</i>,...) • INDEX [<i>index_name</i>] (<i>index_col</i>,...) • UNIQUE [INDEX] [<i>index_name</i>] (<i>index_col</i>,...)

Anweisung	Beschreibung
	<ul style="list-style-type: none"> • FULLTEXT [INDEX] [<i>index_name</i>] (<i>index_col</i>,...) • [CONSTRAINT <i>symbol</i>] FOREIGN KEY <i>index_name</i> (<i>index_col</i>,...) [<i>reference_def</i>] • CHECK (<i>expr</i>) <p><i>type</i> kann einer der Datentypen aus der Tabelle <i>Datentypen</i> (siehe Seite 1080) sein.</p> <p>Für <i>index_col_name</i> kann folgendes eingesetzt werden:</p> <ul style="list-style-type: none"> • <i>col_name</i> [(<i>length</i>)] <p><i>reference_def</i> steht für</p> <ul style="list-style-type: none"> • REFERENCES <i>tbl</i> [(<i>index_col</i>,...)] [MATCH FULL MATCH PARTIAL] [ON DELETE <i>reference_option</i>] [ON UPDATE <i>reference_option</i>] <p><i>reference_option</i> wird ersetzt durch:</p> <ul style="list-style-type: none"> • RESTRICT CASCADE SET NULL NO ACTION SET DEFAULT <p><i>table_options</i> stellt zusätzliche Optionen ein.</p> <p><i>select_statement</i> steht für:</p> <ul style="list-style-type: none"> • [IGNORE REPLACE] SELECT ... (jede gültige SELECT-Anweisung; siehe dort)
ALTER [IGNORE] TABLE <i>tbl</i> <i>alter_spec</i> [, <i>alter_spec</i> ...]	<p>Ändert die Eigenschaften einer Tabelle <i>tbl</i>. <i>alter_spec</i> kann dabei eine der folgenden Werte annehmen:</p> <ul style="list-style-type: none"> • ADD [COLUMN] <i>create_def</i> [FIRST AFTER <i>col</i>] [COLUMN] (<i>create_def</i>, <i>create_def</i>,...) INDEX [<i>index_name</i>] (<i>index_col</i>,...) PRIMARY KEY (<i>index_col</i>,...) UNIQUE [<i>index_name</i>] (<i>index_col</i>,...) FULLTEXT [<i>index_name</i>] (<i>index_col</i>,...) [CONSTRAINT <i>symbol</i>] FOREIGN KEY <i>index_name</i> (<i>index_col</i>,...) [<i>ref_def</i>] • ALTER [COLUMN] <i>col</i> {SET DEFAULT <i>literal</i> DROP DEFAULT} • CHANGE [COLUMN] <i>old_col</i> <i>create_def</i> • MODIFY [COLUMN] <i>create_def</i> • DROP [COLUMN] <i>col_name</i> PRIMARY KEY INDEX <i>index_name</i> • RENAME [TO] <i>new_tbl_name</i> • ORDER BY <i>col</i>

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Anweisung	Beschreibung
	<ul style="list-style-type: none"> • <i>table_options</i> <p>Die Optionen <i>create_def</i> und <i>table_options</i> finden Sie bei CREATE TABLE.</p>
RENAME TABLE <i>old</i> TO <i>new</i>	Benennt eine Tabelle um <i>old</i> in <i>new</i> um
DROP TABLE [IF EXISTS] <i>tbl</i>	Löscht eine Tabelle <i>tbl</i>
OPTIMIZE TABLE <i>tbl</i> , <i>tbl2</i> , ...	Optimiert die Daten von Tabellen <i>tbl</i> , <i>tbl2</i> usw.
CHECK TABLE <i>tbl</i> , <i>option</i>	<p>Prüft eine Tabelle <i>tbl</i>. <i>option</i> kann folgendes sein:</p> <ul style="list-style-type: none"> • QUICK FAST MEDIUM EXTEND CHANGED
BACKUP TABLE <i>tbl</i> TP <i>path</i>	Speichert eine Tabelle <i>tbl</i> auf im Pfad <i>path</i>
RESTORE TABLE <i>tbl</i> FROM <i>path</i>	Stellt eine Tabelle wieder her
ANALYZE TABLE <i>tbl</i>	Analysiert eine Tabelle
REPAIR TABLE <i>tbl</i> [QUICK] [EXTENDED]	Repariert eine Tabelle <i>tbl</i> und gibt eine neue Tabelle mit Informationen über Probleme zurück
DELETE [LOW_PRIORITY] FROM <i>tbl</i> [WHERE <i>where_definition</i>] [LIMIT <i>rows</i>]	Löscht Daten in Tabelle <i>tbl</i>
TRUNCATE TABLE <i>tbl</i>	Leert die Tabelle <i>tbl</i> ohne sie zu löschen
SELECT [STRAIGHT_JOIN] [HIGH_PRIORITY] [DISTINCT DISTINCTROW ALL] <i>select_expression</i> ,... [INTO {OUTFILE DUMPFILE} 'file_name' <i>export_opt</i>] [FROM <i>table_references</i> [WHERE <i>where_def</i>] [GROUP BY { <i>uns_integer</i> <i>col_name</i> <i>formula</i> } [ASC DESC], ...] [HAVING <i>where_definition</i>] [ORDER BY { <i>uns_integer</i> <i>col_name</i> <i>formula</i> } [ASC DESC] ,...] [LIMIT [<i>offset</i> ,] <i>rows</i>] [PROCEDURE <i>procedure_name</i>] [FOR UPDATE LOCK IN SHARE MODE]]	<p>Wählt Daten aus Tabellen aus. <i>select_expression</i> ist dabei ein gültiger Ausdruck aus Variablen, Konstanten und Spaltennamen, der die in den Anhängen B.2 bis B.9 beschriebenen Funktionen enthalten darf.</p> <ul style="list-style-type: none"> • <i>uns_integer</i> ist eine vorzeichenlose Ganzzahl • <i>col_name</i> ist ein Spaltenname • <i>formula</i> ist eine Formel • <i>table_references</i> verweist auf Tabellen

Anweisung	Beschreibung
<code>tbl [CROSS] JOIN tbl2</code> <code>tbl LEFT [OUTER] JOIN tbl2</code> <code>tbl RIGHT [OUTER] JOIN tbl2</code>	Verbindet Tabellen bei Abfragen. Hinter der Verbindung folgt eine Bedingung: • <code>ON condition USING (col, col, ...)</code>
<code>INSERT [INTO] tbl</code> <code>[(col, col, ...)]</code> <code>VALUES</code> <code>(expr, expr, expr, ...)</code>	Fügt Daten in die Tabelle <i>tbl</i> ein.
<code>INSERT [INTO] tbl</code> <code>[(col, col, ...)]</code> <code>SELECT select_option</code>	Fügt Daten einer Auswahlabfrage ein. <i>select_option</i> wird bei SELECT beschrieben.
<code>INSERT DELAYED insert</code>	Entspricht INSERT, wobei jedoch der Client nicht auf die Ausführung der Anweisung warten muss
<code>REPLACE insert</code>	Ersetzt Daten; arbeitet wie INSERT, ersetzt jedoch vorhandene Reihen mit gleichen Daten.
<code>LOAD DATA</code> <code>[LOW_PRIORITY CONCURRENT]</code> <code>[LOCAL] INFILE 'file.txt'</code> <code>[REPLACE IGNORE]</code> <code>INTO TABLE tbl</code> <code>field_description</code>	Lädt Daten aus CSV-Dateien in eine Tabelle. <i>field_description</i> kann folgendes sein: • <code>[FIELDS</code> <code>[TERMINATED BY '\t']</code> <code>[[OPTIONALLY]</code> <code>ENCLOSED BY '']</code> <code>[ESCAPED BY '\\']]]</code> <code>[LINES TERMINATED BY '\n']</code> <code>[IGNORE number LINES]</code> <code>[(col,...)]</code>
<code>UPDATE [LOW_PRIORITY]</code> <code>[IGNORE] tbl</code> <code>SET col1=expr1,</code> <code>[col2=expr2, ...]</code> <code>[WHERE where_definition]</code> <code>[ORDER BY ...] [LIMIT #]</code>	Aktualisiert Daten in einer Tabelle
<code>USE db</code>	Wählt die Datenbank <i>db</i> aus
<code>FLUSH option [, option]</code>	Leert Pufferspeicher. <i>option</i> kann folgendes sein: • <code>HOSTS LOGS PRIVILEGES</code> • <code>TABLES TABLE tbl</code> • <code>STATUS</code>
<code>KILL id</code>	Beendet eine Verbindung zu <i>id</i>
<code>SHOW option</code>	Zeigt Systemdaten an. <i>option</i> kann folgendes sein (<i>wild</i> ist ein in SQL üblicher Platzhalter): • <code>DATABASES [LIKE wild]</code> • <code>[OPEN] TABLES [FROM db] [LIKE wild]</code>

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Anweisung	Beschreibung															
	<ul style="list-style-type: none">• [FULL] COLUMNS FROM <i>tbl</i> [FROM <i>db</i>] [LIKE <i>wild</i>]• INDEX FROM <i>tbl</i> [FROM <i>db</i>]• TABLE STATUS [FROM <i>db</i>] [LIKE <i>wild</i>]• STATUS [LIKE <i>wild</i>] VARIABLES [LIKE <i>wild</i>]• LOGS [FULL] PROCESSLIST• GRANTS FOR <i>user</i> CREATE TABLE <i>tbl</i>• MASTER STATUS MASTER LOGS SLAVE STATUS															
EXPLAIN SELECT <i>select_option</i>	Zeigt das Ausführungsverhalten einer SELECT-Anweisung an															
DESCRIBE <i>tbl</i> { <i>col</i> <i>wild</i> }	Gibt Daten über Spalten <i>col</i> der Tabelle <i>tbl</i> aus.															
BEGIN, COMMIT, ROLLBACK	Transaktionssteuerung															
LOCK TABLES, UNLOCK TABLES	Verriegeln und Freigeben von Tabellen															
SET [OPTION] <i>opt=val</i>	Setzt Parameter															
SET TRANSACTION	Transaktionssteuerung															
GRANT <i>priv_type</i> [(<i>col_list</i>)] [, <i>priv_type</i> [(<i>col_list</i>)]] ON { <i>tbl</i> * *.* <i>db</i> .*} TO <i>user</i> [IDENTIFIED BY ' <i>password</i> '] [, <i>user</i> [IDENTIFIED BY ' <i>password</i> ']] [WITH GRANT OPTION]	Vergibt Rechte an Benutzer <i>user</i> . <i>priv_type</i> kann eines der folgenden Werte sein: <table><tr><td>ALL PRIVILEGES</td><td>FILE</td><td>RELOAD</td></tr><tr><td>ALTER</td><td>INDEX</td><td>SELECT</td></tr><tr><td>CREATE</td><td>INSERT</td><td>SHUTDOWN</td></tr><tr><td>DELETE</td><td>PROCESS</td><td>UPDATE</td></tr><tr><td>DROP</td><td>REFERENCES</td><td>USAGE</td></tr></table>	ALL PRIVILEGES	FILE	RELOAD	ALTER	INDEX	SELECT	CREATE	INSERT	SHUTDOWN	DELETE	PROCESS	UPDATE	DROP	REFERENCES	USAGE
ALL PRIVILEGES	FILE	RELOAD														
ALTER	INDEX	SELECT														
CREATE	INSERT	SHUTDOWN														
DELETE	PROCESS	UPDATE														
DROP	REFERENCES	USAGE														
REVOKE <i>priv_type</i> [(<i>col_list</i>)] [, <i>priv_type</i> [(<i>col_list</i>)]] ON { <i>tbl</i> * *.* <i>db</i> .*} FROM <i>user</i>	Entfernt Rechte für Benutzer <i>user</i> . <i>priv_type</i> finden Sie bei GRANT.															
CREATE [UNIQUE FULLTEXT] INDEX <i>index_name</i> ON <i>tbl</i> (<i>col</i> [(<i>length</i>)],...)	Erzeugt einen neuen Index															
DROP INDEX <i>index_name</i>	Löscht einen Index															
CREATE [AGGREGATE] FUNCTION <i>function_name</i> RETURNS {STRING REAL INT} SONAME <i>libr</i>	Erzeugt eine benutzerdefinierte Funktion															
DROP FUNCTION <i>function_name</i>	Löscht eine benutzerdefinierte Funktion															

C Server-Variablen und Statuscodes

C.1 Server-Variablen

Die folgende Übersicht zeigt alle Servervariablen auf einen Blick. Sie können in PHP direkt darauf zugreifen, da Servervariablen als globale Variablen zur Verfügung stehen, beispielsweise:

```
echo $REMOTE_ADDR;
```

Variablenname	Beschreibung	IIS	Apache
ALL_HTTP	Alle HTTP-Header, die vom Client zum Server gesendet wurden. Das Ergebnis sind Header, die mit HTTP_ beginnen.	•	•
ALL_RAW	Alle HTTP-Header, die vom Client zum Server gesendet wurden. Im Ergebnis werden Header gesendet, die kein Präfix haben.	•	•
APPL_MD_PATH	Gibt den Pfad zur Metabasis der Applikation an (nur IIS/Windows).	•	
APPL_PHYSICAL_PATH	Gibt den physischen Pfad zur Metabasis der Applikation an (nur IIS/Windows).	•	
AUTH_PASSWORD	Das Kennwort einer Autorisierung, wenn es im Kennwortfeld des Browsers eingegeben wurde (nur IIS/Windows).	•	
AUTH_TYPE	Art der Autorisierung, wenn Nutzer Zugriff auf ein geschütztes Dokument haben möchten (nur IIS/Windows).	•	
AUTH_NAME	Name des Nutzers bei Eingabe in das Kennwortfeld des Browsers.	•	
CERT_COOKIE	Eindeutige ID eines Clientzertifikats.	•	
CERT_FLAGS	Flag des Clientzertifikats, Bit 0 ist 1, wenn das Clientzertifikat vorhanden ist, Bit 1 ist 1, wenn das Clientzertifikat nicht überprüft wurde.	•	
CERT_ISSUER	Das Issuer (Herausgeber)-Feld des Clientzertifikats.	•	
CERT_KEYSIZE	Bitzahl bei einer SSL-Verbindung.	•	
CERT_SECRETKEYSIZE	Anzahl der Bits eines privaten Zertifikatschlüssels.	•	
CERT_SERIALNUMBER	Die Seriennummer des Zertifikats.	•	

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

Variablenname	Beschreibung	IIS	Apache
CERT_SERVER_ISSUER	Das Issuer (Herausgeber)-Feld des Serverzertifikats (Issuer-Feld).	•	
CERT_SERVER_SUBJECT	Beschreibung des Zertifikats (Server).	•	
CERT_SUBJECT	Beschreibung des Zertifikats (Client).	•	
CONTENT_LENGTH	Länge des zu sendenden Inhalts.	•	•
CONTENT_TYPE	Art des Inhalts (MIME-Type)	•	•
DATE_LOCAL	Datum/Uhrzeit des Servers		•
DATE_GMT	Datum/Uhrzeit des Servers in Zone GMT		•
DOCUMENT_NAME	Name des ausführenden Dokuments		•
DOCUMENT_ROOT	Pfad zum Dokument ohne Dateiname		•
GATEWAY_INTERFACE	Art des Interface, das der Server benutzt.	•	•
HTTPS	Ist ON, wenn der Server SSL benutzt.	•	•
HTTPS_KEYSIZE	Schlüssellänge der HTTPS-Verbindung (40bit, 128bit...)	•	•
HTTPS_SECRETKEYSIZE	Schlüssellänge bei privaten Zertifikaten	•	•
HTTPS_SERVER_ISSUER	Issuer-Feld des Serverzertifikats bei sicherer Übertragung	•	•
HTTPS_SERVER_SUBJECT	Beschreibung	•	•
INSTANCE_ID	ID-Nummer der Instanz (nur IIS)	•	
INSTANCE_META_PATH	Der Metabasispfad (nur IIS)	•	
LOCAL_ADDR	Die in der Anforderung benutzte Serveradresse	•	•
LOGON_USER	Ein Nutzerkonto	•	
PATH_INFO	Pfadinformation für den Client	•	•
PATH_TRANSLATED	Übertragung der Pfadinformation ins physische Format	•	•
QUERY_STRING	Inhalt des Querystrings (Parameter-URL)	•	•
REMOTE_ADDR	Die IP-Adresse des Nutzers	•	•
REMOTE_HOST	Name des Computers des Nutzers	•	•
REQUEST_METHOD	Die Methode der Datenübertragung eines Formulars. Kann GET, PUT oder POST sein.	•	•
REQUEST_URI	URI der Anforderung		•

Variablenname	Beschreibung	IIS	Apache
SCRIPT_FILENAME	Name eines Skripts		•
SCRIPT_NAME	Name eines Skripts	•	
SERVER_NAME	Der Hostname des Servers, eine DNS- oder IP-Adresse	•	•
SERVER_PORT	Port, der vom Server benutzt wird (normalerweise 80)	•	•
SERVER_PORT_SECURE	Port, der bei sicherer Übertragung benutzt wird (Standard: 443)	•	•
SERVER_PROTOCOL	Das verwendete Protokoll und die Version (z.B.: HTTP1.1)	•	•
SERVER_SIGNATURE	Signatur des Servers (einstellbare Info)		•
SERVER_SOFTWARE	Der Name und die Version der auf dem Server laufenden Software	•	•

Wichtige Anmerkung

Nicht alle Variablen stehen in allen Umgebungen zur Verfügung. Einige Variablen des Apache sind nur verfügbar, wenn der Server entsprechend eingerichtet wurde. Beispielsweise muss SERVER_SIGNATURE bei der Konfiguration eingegeben worden sein; dieser Wert ist nicht standardmäßig verfügbar. Testen Sie Skripte vor der Auslieferung auf mehreren Servern, wenn diese auf bestimmte Servervariablen angewiesen sind!

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

C.2 HTTP-Statuscodes

In der HTTP-Referenz finden Sie alle Codes der Version 1.1 in fünf Gruppen. In den jeweiligen Einleitungen wird auf Unterschiede zur Version 1.0 eingegangen. Die Liste basiert auf der Übersetzung des vom W3C am 15.5.1998 herausgegebenen Papiers.

Übersicht der Codes

Informationscodes

100 Continue

101 Switching Protocols

Erfolgsmeldungen

200 OK

201 Created

202 Accepted

203 Non-Authoritative Information

204 No Content

205 Reset Content

206 Partial Content

Umleitungsmeldungen

300 Multiple Choices

301 Moved Permanently

302 Moved Temporarily

303 See Other

304 Not Modified

305 Use Proxy

Client-Fehler

400 Bad Request

401 Unauthorized

402 Payment Required

403 Forbidden

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

- 404 Not Found
- 405 Method Not Allowed
- 406 Not Acceptable
- 407 Proxy Authentication Required
- 408 Request Timeout
- 409 Conflict
- 410 Gone
- 411 Length Required
- 412 Precondition Failed
- 413 Request Entity Too Large
- 414 Request-URL Too Long
- 415 Unsupported Media Type

Server-Fehler

- 500 Internal Server Error
- 501 Not Implemented
- 502 Bad Gateway
- 503 Service Unavailable
- 504 Gateway Timeout
- 505 HTTP Version Not Supported

C.3 Die Codes im Detail

Informationscodes – Informational Codes 1xx

Diese Gruppe enthält einfache, informelle Antworten, die nur aus der Statuszeile und optional verfügbaren Headern bestehen. Der Code wird durch eine Leerzeile abgeschlossen. 1xx-Codes sind unter HTTP 1.0 nicht explizit definiert, sodass Server, die einen HTTP 1.0-Client bedienen, diese Codes nicht unbedingt ausgeben müssen.

100 Continue

Der Client kann die Anfragen fortsetzen. Damit wird der Client informiert, dass die bisherigen Anfragen empfangen wurden, die Antwort bearbeitet wird, möglicherweise aber noch zusätzliche Informationen nötig sind, um korrekt zu antworten. Der Client sollte unmittelbar danach den Rest der Anfrage senden oder diese Meldung ignorieren, wenn

er bereits fertig ist. Der Server *muss* eine abschließende Meldung senden, wenn die Anfrage komplett ist.

101 Switching Protocols

Der Server hat eine Anfrage nach dem Wechsel des Applikationsprotokolls verstanden und ist mit dem Wechsel einverstanden. Der Server wechselt zu den Protokollen, die im Antwort-Header-Feld definiert wurden. Diese Information steht unmittelbar hinter der Leerzeile, die die Meldung 101 abschließt. Die Änderung des Protokolls sollte nur dann benutzt werden, wenn sich für die folgende Übertragung tatsächlich ein Vorteil ergibt.

Erfolgsmeldungen – Successful 2xx

Diese Gruppe enthält Erfolgsmeldungen, die im Anschluss an eine erfolgreiche Transaktion gesendet werden. Erfolgreiche Transaktionen resultieren aus dem Verstehen, Akzeptieren oder Empfangen von Daten.

200 OK

Die Anfrage wurde erfolgreich bearbeitet. Die Information, die zusätzlich zur Anfrage gesendet wurde, hängt von der Art der Anfrage ab.

- Nach GET werden die Daten im Body mitgeliefert, die angefordert wurden.
- Nach HEAD werden die Headerdaten geliefert.
- Mit POST wird eine Beschreibung oder der Inhalt einer Aktion gesendet.
- Bei TRACE wird die Anfrage geliefert, die der Server verstanden hat.

201 Created

Die Anfrage wurde ausgeführt und führte zu einer neuen Ressource (Datenquelle), die erfolgreich erzeugt wurde. Die neu erzeugte Ressource kann durch verschiedene Informationen, die im Body der Meldung stehen, beschrieben werden. Antworten können eine oder mehrere URLs sein. Wenn der Server 201 sendet, ist die Ressource bereits erzeugt. Dauert die Aktion an, auch wenn sie Erfolg versprechend verläuft, sendet der Server die Meldung 202.

202 Accepted

Die Anfrage wurde verstanden und akzeptiert, aber noch nicht abschließend ausgeführt. Diese Meldung sollte nicht eingesetzt werden, um mehrstufige Transaktionen zu überwachen. Sie erlaubt es aber beispielsweise Batchprozessen, automatisch und kontinuierlich zu laufen, ohne dass der Server jede Aktion sofort ausführt. Im Body sollte eine Information stehen, die über den Verlauf der Aktion berichtet (erwartete Ausführungszeit, Status, Zwischenbericht).

203 Non-Authoritative Information

Der Header enthält zusätzliche (Meta-)Informationen, die nicht vom Webserver selbst erzeugt wurden, sondern von einer Software, die von einem weiteren Anbieter auf dem System eingesetzt wird (Third-Party-Software). Der Einsatz macht nur Sinn, wenn die Meldung 200 unzureichend ist.

204 No Content

Der Server hat die Anfrage verstanden, es gibt aber nichts zu senden. Damit wird der Client informiert, dass er seine Sicht der Inhalte (HTML-Seite, Formular) nicht ändern muss. Die Meldung wird mit der Leerzeile beendet.

205 Reset Content

Die Anfrage wurde erfolgreich ausgeführt, und der Client sollte nun das Dokument neu aufbauen. Bei Formularen sollte der Client die Felder löschen und die erneute Eingabe der Daten ermöglichen.

206 Partial Content

Der Server beantwortet die Anforderung eines Teils der Daten mit GET. Die Header müssen Informationen über die Größe und Art der Daten enthalten (Range- und Content-Range-Header).

Umleitungsmeldungen – Redirection 3xx

Diese Klasse beschreibt Statuscodes, die weitere Aktionen des Clients verlangen.

300 Multiple Choices

Die Ressource enthält Informationen über mehrere Datenquellen an bestimmten Orten. Der Client wird aufgefordert, eine entsprechende Wahl zu treffen. Dieser Standard spezifiziert keine Vorschriften, die die Art und Weise der Transaktion beschreiben.

301 Moved Permanently

Die angeforderte Ressource wurde permanent an einen anderen Ort verschoben. Der Body der Meldung enthält die neue URL. Wenn die Anfrage GET oder HEAD war, sollte der Client dem neuen Link automatisch folgen, ansonsten erst nach einer Bestätigung durch den Nutzer.

302 Moved Temporarily

Die angeforderte Ressource wurde nur zeitweilig verschoben. Die neue URL wird gesendet. Wenn die Anfrage GET oder HEAD war, sollte der Client dem neuen Link automatisch folgen, ansonsten erst nach einer Bestätigung durch den Nutzer.

303 See Other

Die Anfrage kann nur von einer anderen Stelle der Ressource angefordert werden. Der Client sollte dazu die GET-Methoden verwenden. Die Meldung bedeutet nicht, dass die Ressource verschoben wurde.

304 Not Modified

Das angeforderte Dokument hat sich seit der letzten Anforderung nicht geändert. Die Meldung enthält keine Daten im Body, dafür die folgenden Header-Felder:

- Date (Datum),
- Tag und/oder Content-Location, wenn die Meldung 200 bereits erfolgte,
- Expires (Verfallsdatum), Cache-Control (Speicher) und/oder Vary (Varianz), wenn es die Feldwerte erfordern.

305 Use Proxy

Die Ressource, die angefordert wurde, sollte aus dem Cache des Proxys genommen werden, der im Location-Feld angegeben wurde. Der Client sollte die Anfrage erneut unter Nutzung des Proxy starten.

Client-Fehler – Client Error 4xx

Fehler auf der Client-Seite. Der Server vermutet, dass die Ursache des Fehlers auf der Seite des Clients zu suchen ist.

400 Bad Request

Die Anfrage wurde nicht verstanden. Der Client sollte die Anfrage nicht wiederholen, ohne dass Änderungen vorgenommen wurden.

401 Unauthorized

Die Anfrage erforderte eine Authentifizierung des Nutzers. Die Antwort muss ein WWW-Authenticate-Header-Feld beinhalten.

402 Payment Required

Bezahlung erforderlich. Dieser Code ist eine Meldung für die Zukunft.

403 Forbidden

Die Anfrage wurde verstanden, die Ausführung ist aber verboten. Es ist nicht das Problem einer mangelhaften Autorisierung, sondern die Anfrage ist grundsätzlich nicht erlaubt. Der Client sollte die Anfrage nie wiederholen.

404 Not Found

Die angeforderte Ressource wurde nicht gefunden. Meist bei Schreibfehlern oder gelöschten Dateien. Wurde die Ressource ohne Weiterleitung verschoben, sollte die Meldung 410 benutzt werden.

405 Method Not Allowed

Die Methode, die zur Ausführung spezifiziert wurde, ist für diese Ressource nicht erlaubt.

406 Not Acceptable

Es wurden bestimmte Charakteristiken zur Selektierung der Ressource gesendet und keine dieser Angaben führte zu einer Antwort.

407 Proxy Authentication Required

Dieser Code ähnelt der Meldung 401 (Unauthorized), zeigt aber an, dass sich der Client zuerst am Proxy autorisieren muss.

408 Request Timeout

Der Client hat innerhalb der erwarteten oder zulässigen Wartezeit nicht reagiert.

409 Conflict

Durch den aktuellen Status der Ressource konnte die Anfrage nicht ausgeführt werden. Dieser Code sollte nur gesendet werden, wenn der Client in der Lage ist, darauf zu reagieren, oder der Fehler beim Client lag (Ressource falsch). Solche Zustände können bei der Verwendung der PUT-Methoden mit Programmen fremder Hersteller auftreten.

410 Gone

Die Ressource ist nicht mehr verfügbar und ein neuer Platz ist nicht bekannt.

411 Length Required

Bei der Anforderung fehlte die Längenangabe (Bytezahl) der Ressource.

412 Precondition Failed

Die in den Request Headern übergebene Precondition ist falsch oder wurde nicht verstanden.

413 Request Entity Too Large

Der Server meint, dass die angeforderte Ressource zu groß ist, um übertragen zu werden.

414 Request-URL Too Long

Die URL war so lang, dass der Server sie nicht mehr bearbeiten konnte. Kommt vor allem bei sehr langen GET- und POST-Methoden vor.

415 Unsupported Media Type

Der Typ der Ressource und der in der Anfrage genannte Typ stimmen nicht überein. Der Typ wird nicht unterstützt.

Server-Fehler – Server Error 5xx

Diese Fehler werden angezeigt, wenn der Server selbst Probleme hat, eine Anfrage auszuführen. Die zusätzlich gelieferten Informationen im Body sollte der Client anzeigen.

500 Internal Server Error

Interner Server-Fehler. Es kann sich um hardwarebedingte Fehlfunktionen handeln (Speichermangel, Plattenfehler usw.).

501 Not Implemented

Die Anfrage konnte nicht nur für die benannte Ressource nicht ausgeführt werden, sondern wird grundsätzlich nicht unterstützt.

502 Bad Gateway

Wenn der Server als Proxy oder Gateway arbeitet, zeigt er damit an, dass der übergeordnete Server fehlerhafte Daten geliefert hat.

503 Service Unavailable

Der Server ist zeitweilig überlastet und kann keine weiteren Anfragen bearbeiten. Die Existenz dieses Codes bedeutet nicht, dass der Server bei jeder Überlastung wirklich noch auf neue Anfragen reagiert.

504 Gateway Timeout

Wenn der Server als Proxy oder Gateway arbeitet, zeigt er damit an, dass der übergeordnete Server keine Daten innerhalb der erwarteten Zeitspanne geliefert hat.

505 HTTP Version Not Supported

Das angeforderte Protokoll wird nicht unterstützt. Die unterstützten Protokolle werden im Body gesendet.

D Index

D.1 Erläuterungen zum Index

Generell erhebt dieser Index trotz beachtlichen Umfangs keinen Anspruch auf Vollständigkeit. Es wurde versucht, jeweils die für ein Stichwort relevante Stelle anzugeben, nicht jedoch ein Verweis auf jedes Vorkommen. Stattdessen finden Sie im Buch Querverweise auf weiterführende Stellen, die mit dem Symbol ➡ gekennzeichnet sind.

Bedeutung der Seitenzahlen

Im Index finden Sie normale, **fette** und *kursive* Seitenzahlen. Alle Funktionen, die PHP mitbringt, sowie die Kontrollstrukturen und Befehle sind fett: `array()` 234. Alle Konstanten, die vordefiniert sind, werden kursiv gekennzeichnet: *\$GLOBALS* 123. Alle übrigen Begriffe sind normal bezeichnet, auch SQL-Kommandos (siehe unten).

Funktionen und Kontrollstrukturen

Der Index enthält alle Elemente aus dem Text. Zum schnellen Auffinden der Funktionen sind folgende Kennzeichnungen zu beachten:

- Funktionen sind durch zwei runde Klammern gekennzeichnet, beispielsweise `nl2br()`.
- Der Index enthält keine Verweise auf die Referenz, da diese bereits zwei Suchoptionen bietet: die alphabetische Funktionsliste und die Sortierung nach Funktionsgruppen.

SQL-Kommandos

Im Index sind auch SQL-Anweisungen aufgeführt. Diese sind generell, wie auch im Buch, in Großbuchstaben geschrieben:

- ALTER TABLE

Abkürzungen

Abkürzungen sind in der Regel großgeschrieben. Zur Abgrenzung zu den SQL-Anweisungen sind keine direkten Verweise auf Abkürzungen aufgeführt. Stattdessen wird immer auf die ausgeschriebene Version verwiesen, oft ist das auch schon die Antwort auf die Frage:

- CVS *Siehe* Concurrent Version System

D.2 Globaler Index

\$

\$\$vari 159
 \${LONGDATE} 775
 \${LONGTIME} 775
 \${MAIN} 776
 \${SESSIONID} 775
 \${SHORTDATE} 775
 \${SHORTTIME} 775
 \${TEMPLATE} 775
 \${USERID} 775
 \${USERNAME} 775
 \$GLOBALS 158
 \$HTTP_COOKIE_VARS 332
 \$HTTP_POST_VARS 317
 \$PHP_AUTH_PW 408
 \$PHP_AUTH_USER 408
 \$PHP_ERRORMSG 274
 \$PHP_SELF 339
 \$QUERY_STRING 321
 \$this 257

(

(array) 163
 (double) 163
 (int) 163
 (integer) 163
 (object) 163
 (real) 163
 (string) 163

—

__FILE__ 169
 __LINE__ 169

<

<LS
 DATA> 785
 DATANEXT> 785
 DATAWHILE> 785
 ELSE> 792
 ENCODING> 798
 FOLLOW> 778
 IF> 792
 IFEMPTY> 793
 IFNOT> 792
 PROCESS> 794
 SET> 776
 SETGLOBAL> 776
 STOP> 793
 UNSET> 776
 UNSETGLOBAL> 776

A

Abbruchsteuerung 271
 Aborted 334
 abs() 204
 Access 524
 Datenbank anlegen 524
 Access 2000 570
 acos() 204
 addcslashes() 189
 Addition 170
 addslashes() 189, 197

ALL 493
ALTER TABLE 486
American Registry for Internet Numbers 66
and 231
AND 494
ANSI SQL92 463
Apache
 Konfiguration unter Windows 108
APNIC *Siehe* Asia-Pacific Network Information Center
ARIN *Siehe* American Registry for Internet Numbers
ARPANET 53
array() 174
array_compact() 181
array_count_values() 179
array_extract() 181
array_flip() 179
array_keys() 185
array_merge() 186
array_merge_recursive() 179
array_multisort() 179
array_pad() 186
array_pop() 180
array_push() 180
array_reverse() 180, 186
array_shift() 180
array_slice() 180
array_splice() 180
array_unique() 181
array_unshift() 180
array_values() 185
array_walk() 182
Arrayfunktionen 177
 Übersicht 177

Arrays 173
 Assoziative 175
 Indizierte 173
 mehrdimensionale 176
arsort() 188
as *Siehe* foreach
ASC 491
Asia-Pacific Network Information Center 66
asin() 204
asort() 188
ASP-Style 148
atan() 204
atan2() 204
Aufbau des Internet 49
Aufzählungstyp 482
Ausführungssteuerung 336
Authentifizierung 401
 Apache-Modul 407
AUTO_INCREMENT 485
AVG 501

B

Back Reference *Siehe* Reguläre Ausdrücke
base_convert() 205
Base64 95
base64_decode() 229
base64_encode() 229
basename() 352
Bedingungen 231
Berliner Mauer 841
BETWEEN 495
Beziehungen 473
Bilder 412
Bilderzeugung 411

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

- Bildformate
 - GIF 412
 - JPEG 412
 - PNG 413
- Bildfunktionen 414
 - Anwendungsbeispiel 420
- Binary, 8-Bit, Raw 98
- bindec() 205
- BinHex 96
- Bitoperatoren 171
 - NICHT 171
 - ODER 171
 - UND 171
- blat 647
- Blöcke 230
- BOO 98
- break
 - for 242
 - switch 237
 - while 239
- Broadcast *Siehe* Broadcast-Adresse
- Broadcast-Adresse 65
- BTOA 98
- ByRef 247
- ByVal 247

C

- call_user_method() 260
- call_user_method_array() 260
- case 236
- ccTLD *Siehe* Country Code Top-Level Domain
- ceil() 205
- Cforce (Editor) 750
- CGI 47
- checkdate() 209

- checkdnsrr() 635
- chgrp() 403
- chmod() 403
- chop() 194
- chown() 403
- chr() 197
- chunk_split() 191
- class 256
- class_exists() 263
- clearstatcache() 365
- Client Uniform Resource Locator 386
- Code-Konventionen 278
- Code-strukturierung 281
- COM 268
- COM-Objekte 268
- connection_aborted() 335
- connection_status() 336
- connection_timeout() 335
- continue
 - for 242
 - while 240
- convert_cyr_string() 191
- Cookie setzen 230
- Cookies
 - Definition 325
 - Einführung 324
 - Namenskonflikte 332
 - Spezifikation 325
 - und JavaScript 333
- copy() 366
- CORBA 268
- cos() 204
- COUNT 500
- COUNT(*) 501
- Counter 439
 - bildbasiert 440

textbasiert 439
Country Code Top-Level Domain 54
CREATE DATABASE 483
CREATE TABLE 483
crypt() 189
CURL *Siehe* Client Uniform Resource Locator
curl_close 387
curl_errno 387
curl_error 387
curl_getinfo 388
curl_init 387
curl_setopt 389
current() 184

D

Data Control Language 474
Data Definition Language 474
Data Manipulation Language 474
Data Source Name 520
date() 209
Dateien einschließen 150
Dateierweiterungen 147
Dateisystem 350
 Zugriffsrechte 377
Dateiupload 372, 431
 Größe begrenzen 376
 Mehrere Dateien 377
 PUT-Methode 377
Daten weiterreichen 318
Datenbank verbinden 534
Datenbankcursor 545
Datenbanken
 Einführung in SQL 487
 löschen 485
 Speicherbedarf 480
Datenbankhandle 535
Datenbankprogrammierung 531
Datenbankprojekt 552
Datenbanktechnik 455
Datenbankzugriff mit Access 524
Datenbankzugriffe 828
Datennormalisierung 471
Datentyp
 array 161
 double 161
 int 161
 integer 161
 object 161
 real 161
 string 161
Datentypen 161
 ENUM 482
 numerische 481
 SET 482
 Zeichenkettentypen 482
Datumsberechnungen 212
Datumsfunktionen 209
DCL *Siehe* Data Control Language
DCOM 268
DDL *Siehe* Data Definition Language
Debugging 286
decbin() 205
dechex() 205
decoct() 205
default 237
define() 168
defined() 168
Dekrement 170
DELETE 490
delete() 367
Department of Defense 53

V

S

I

1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E

- DESC 491
- die 272
- dir 368
- dir->close 368
- dir->handle 368
- dir->path 368
- dir->read 368
- dir->rewind 368
- diskfreespace() 352
- DISTINCT 492
- Dithering 412
- Division 170
- DML *Siehe* Data Manipulation Language
- DNS-Funktionen 634
- do ... while 240
- DoD *Siehe* Department of Defense
- DoubleClick 329
- doubleval() 165
- Draft 58
- DROP DATABASE 485
- DROP TABLE 485
- DSN *Siehe* Data Source Name
- DTD *Siehe* XML
- Dynamische Variablen mit Arrays 160
- E**
- E_CORE_ERROR 273
- E_CORE_WARNING 273
- E_ERROR 170, 273
- E_NOTICE 170, 273
- E_PARSE 170, 273
- E_WARNING 170, 273
- each() 183
- easter_date() 208
- easter_days() 208
- echo 152
- Editoren 749
 - für Unix 750
 - für Windows 749
- EditPlus (Editor) 749
- Electronic-Mail 51
- else 234
- elseif 234
- E-Mail *Siehe* Electronic Mail
 - unter Windows 646
- E-Mail-Clients 99
- E-Mail-Header 100
- E-Mail-Programmierung 635
- E-Mail-Protokolle
 - POP3 87
- empty() 166
- end() 184
- endfor 243
- endif 235
- endwhile 241
- ereg() 227
- ereg_replace() 227, 694
- eregi() 227
- eregi_replace() 227
- Erforderliche Kenntnisse 47
- error_reporting() 273
- Erste Schritte 147
- Erweiterte OOP-Syntax 261
- Erweiterte Syntax 222
- Escape-Zeichen
 - störende 320
- ESMTP *Siehe* Extended Simple Mail Transfer Protocol
- eval() 296
- EXISTS 495

exit 272
exp() 204
Experimental 57
EXPLAIN 513
explode() 191, 195, 323
extends 255, 267
Extensible Markup Language 697
extract() 295
Extranet 52

F

Fakultät 425
FALSE 169
FastTemplate 770
fclose() 358
Fehler auswerten 274
Fehler vermeiden 278
Fehler verstecken 273
Fehleranzeige modifizieren 274
Fehlerbehandlung 271
Fehlerbehandlungsroutinen 286
Fehlerkonzept 272
Fehlersuche 285
Feldinformationen 548
feof() 361
fgetcsv() 352
fgets() 358
fgetss() 352
FIELDS 510
File Transfer Protocol 51, 77
 Authentifizierung 79
 Datenstruktur 80
 Kommandos 78
 Statuscodes 81
file() 356

file_exists() 352
fileatime() 352
filectime() 352
File-DSN 521
filemtime() 362
fileowner() 362
fileperms 354
fileperms() 363
filesize() 363
filetype() 364
Fließkommazahl 162
flock() 354
floor() 205
FLUSH 511
flush() 190
fopen() 357
for 241
 Alternative Syntax 243
foreach 244
Formatierung 281
Formatierungen *Siehe*
 Zeichenkettenformatierungen
Formatierzeichenkette 203
Formulare 301
 erzeugen mit PHP 315
 HTML 303
 JavaScript 308
 Mehrfachauswahl 314
Formularelemente auf Existenz testen 307
Formularmanagement 777
fputs() 358
fread() 358
Freetype 411
frenchtojd() 207
frewind() 362

V

S

I

1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E

- fseek() 362
- fsockopen() 383, 631
- ftell() 362
- FTP *Siehe* File Transfer Protocol *Siehe* File Transfer Protocol
 - Passiver Modus 81
 - Transfer-Modus 81
- ftp_cdup() 393
- ftp_connect() 393
- ftp_delete() 394
- ftp_fget() 394
- ftp_fput() 394
- ftp_get() 394
- ftp_mdtm() 394
- ftp_mkdir() 393
- ftp_nlist() 394
- ftp_pasv() 394
- ftp_put() 394
- ftp_pwd() 393
- ftp_quit() 394
- ftp_rename() 394
- ftp_rmdir() 393
- ftp_size() 394
- ftp_systype() 394
- FTP-Adresse mit Kennwort 382
- FTP-Funktionen 393
- func_get_arg() 250
- func_get_args() 250
- func_num_args() 250
- function() 246
- function_exists() 285
- Funktionen 245
 - Erweiterte Syntax 253
 - globale Variablen 247
 - Optionale Parameter 248
 - Referenzen 247
 - Rekursion 251
 - Rückgabe mehrerer Werte 251
 - Variable Argumentelisten 249
- Funktionsargumente 246
- Funktionsdefinition 246
- fwrite() 358
- G**
 - Ganzzahl 161
 - GD-Bibliothek 411
 - General-Header-Fields 75
 - Geschachtelte Nachrichten 94
 - GET 302
 - get_class() 260, 262
 - get_class_methods() 260, 263
 - get_class_vars() 263
 - get_declared_classes() 263
 - get_html_translation_table 296
 - get_meta_tags() 190
 - get_method() 262
 - get_object_vars() 263
 - get_parent_class() 262
 - get_resource_type() 166
 - getdate() 209
 - gethostbyname() 635
 - getrandmax() 207
 - gettimeofday() 212
 - gettype() 165
 - global 158
 - Globale Servervariablen 317
 - Glossar 849
 - gmdate() 209
 - gmmktime() 212
 - Gopher 52
 - gpc_order 332

GRANT 515

Greedy *Siehe* Reguläre Ausdrücke

GROUP BY 493

Grundbegriffe 49

Gateways 49

Internet 49

PoP 49

Provider 49

Router 49

Gültigkeitsbereiche 157

H

Hardware für PHP 106

HAVING 494

header() 230, 632

heredoc 153

hexdec() 205

Hilfsfunktionen 294

Homesite (Editor) 749

htaccess 409

HTML 50

htmlentities() 199

HTML-Formularelemente 304

HTML-Funktionen 199

HTML-Mail 97

htmlspecialchars() 199

HTTP *Siehe* Hypertext Transfer Protocol

HTTP-Funktionen 229

HTTP-Verbindungen 381

Hypertext Transfer Protocol 72

Message-Header 75

Statuscodes 74

I

IANA *Siehe* Internet Assigned Numbers Authority *Siehe* Internet Assigned Number Authority

ICANN *Siehe* Internet Corporation for Assigned Names and Numbers

Icons und Symbole 29

Identitätsoperatoren 232

if 233

Alternative Syntax 235

ignore_user_abort() 334

imagecreate() 419

imap_header() 652

imap_num_msg() 652

IMAP-Funktionen 646

implode() 191, 195

IN 495

in_array() 181

include() 150

Inkrement 170

INSERT 489

Installation

Linux 114

PHP 3 109

PHP 4 109

PHP als ISAPI-Modul 110

Windows 107, 110

Windows (PWS) 112

Windows (Xitami) 113

Interaktive Webseiten 299

International Organization for Standardization 54

Internet Assigned Number Authority 54

Internet Assigned Numbers Authority 66

Internet Corporation for Assigned Names and Numbers 53

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

- Internet Protocol
 - Broadcast 62
 - Datagramm 59
 - Fragmentierung 61
 - Multicast 62
 - Subnetze 63
 - Unicast 62
- Internet Protocol Suite 53
- Internet Protocol Version 4 63
- Internet Protocol Version 6 63
- Internet Relay Chat 51
- INTERVAL 496
- Intranet 52
- intval() 165
- IP 71
- IP-Adressen
 - Adressversionen 63
- IPS *Siehe* Internet Protocol Suite
- IPv4 *Siehe* Internet Protocol Version 4
- IPv6 *Siehe* Internet Protocol Version 6
- IRC *Siehe* Internet Relay Chat
- is_array() 165
- is_dir() 362
- is_double() 165
- is_executable() 363
- is_file() 362
- is_integer() 165
- is_link() 362
- is_null() 166
- is_object() 165
- is_readable() 363
- is_resource() 166
- is_scalar() 166
- is_string() 165
- is_subclass_of() 260, 263

- is_writeable() 363
- ISAPI 48
- ISO *Siehe* International Organization for Standardization
- ISO/OSI-Referenzmodell 54
- isset() 166
- IT[X] 770

J

- James Clark *Siehe* XML
- jdtofrrench() 207
- jdtogregorian() 207
- jdtojulian() 207
- jewishtojd() 207
- JOIN 508
- join() 191
- Joint Picture Expert Group *Siehe* JPEG
- JPEG 412
- Jüdischer Kalender *Siehe* Kalenderfunktionen
- Julianischen Kalender *Siehe* Kalenderfunktionen
- juliantojd() 207

K

- kalenderfunktionen
 - jdtojewish() 207
- Kalenderfunktionen 207
- key() 185
- KILL 512
- Klasse-A-Netz 64
- Klasse-B-Netz 64
- Klasse-C-Netz 65
- Klasse-D-Netz 65
- Kommentare 154
 - HTML 155

PHP 155
Kommentarkonventionen 278
komplexe Datenstrukturen 174
Konfiguration
 Groß- und Kleinschreibung 120
Konfigurationsdatei 120
Konstanten 168
 mathematische 206
 vordefinierte 169
ksort() **188**

L

LAMP *Siehe* Linux Apache MySQL PHP
Leerzeichen entfernen 194
LEFT JOIN 509
LEFT OUTER JOIN 509
Lesbarkeit 282
levenshtein() **193**
LIKE 492
LIMIT 494
link() **355**
linkinfo() **355**
Linux 114
Linux Apache PHP 114
list() **182**
LOAD DATA 509
log() **204**
log10() **205**
Logische Operatoren 172, 231
 Assoziativität 172
Loopback 65
lstat() **355**
ltrim() **194**

M

M_1_PI 206
M_2_PI 206
M_2_SQRTPI 206
M_E 206
M_LN10 206
M_LN2 206
M_LOG10E 206
M_LOG2E 206
M_PI 206
M_PI_2 206
M_PI_4 206
M_SQRT1_2 206
M_SQRT2 206
mail() **649**
Mathematische Funktionen 204
Mauerfotos 841
MAX 501
max() **204**
MAX_FILE_SIZE 376, 377
md5() 189, **341**
MD5-Verschlüsselung 341
Mehrspaltige Ausgabe 790
metaphone() **193**
method_exists() **260, 262**
Microsoft Guidelines 278
Microsoft SQL Server 521
microtime() **212**
MIME 90
MIME-Typen 91
MIN 501
min() **204**
Mini-Debugger 287
mktime() **213**

V

S

I

1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E

- Modulus 170
 - MS Access 519
 - MS Excel 519
 - mt_getrandmax() 207
 - mt_rand() 207
 - mt_srand() 207
 - mt-Funktionen *Siehe* Zufallszahlen-Funktionen
 - Multiplikation 170
 - MyODBC 584
 - MySQL 457
 - Aggregat-Funktionen 500
 - Besonderheiten 478
 - Dateierweiterungen 468
 - Datensicherung 509
 - Datentypen 478
 - Datums- und Zeitfunktionen 504
 - Datumsarithmetik 505
 - Datumsformatierungen 506
 - Erweiterungen 464
 - Fehlende Funktionen 465
 - Gruppierungen 493
 - Installation unter Windows 457
 - Literale 475
 - Logische Funktionen 499
 - Mathematische Funktionen 498, 1081
 - Namenskonventionen 477
 - Operatoren 497
 - phpMyAdmin 586
 - Portierbarkeit 466
 - Reguläre Ausdrücke 496
 - Sicherheit 514
 - Systemfunktionen 501, 511
 - Technische Daten 467
 - über Access bedienen 582
 - Usertabellen 517
 - versus Standard-SQL 463
 - Zeichenkettenfunktionen 502
 - Zugriffsrechte 514
 - mysql_affected_rows() 541
 - mysql_close() 545
 - mysql_connect() 536
 - mysql_create_db() 543
 - mysql_data_seek() 546
 - mysql_drop_db() 543
 - mysql_fetch_array() 538, 546
 - mysql_fetch_fields() 548
 - mysql_fetch_length() 548
 - mysql_fetch_object() 546
 - mysql_fetch_row() 546
 - mysql_field_len() 550
 - mysql_field_name() 550
 - mysql_field_seek() 549
 - mysql_field_table() 551
 - mysql_field_type() 550
 - mysql_insert_id() 540
 - mysql_list_dbs() 543
 - mysql_listdbs() 552
 - mysql_listtables() 552
 - mysql_num_fields 546
 - mysql_num_rows() 537, 546
 - mysql_pconnect() 545
 - mysql_query() 537
 - mysql_select_db() 536
 - mysql_tablename() 552
 - MySQL-Funktionen 531
 - Fehlersuche 534
 - MySQL-Monitor 459
 - MySQL-ODBC 527
 - MySQL-ODBC-Treiber *Siehe* MyODBC
- N**
- Namenskonventionen 278

- Navigation
 - seitenweise 787
- Nedit (Editor) 750
- Network News Transfer Protocol 104
- Netzwerkklassen 64
- Netzwerkprogrammierung 631
- new 258, 259, 1085
- News
 - Threadverfolgung 106
- Newsserver 650
- next() 183
- nl2br() 200
- NNTP *Siehe* Newsserver *Siehe* Network News Transfer Protocol
- Normal 334
- Normalisierungsregeln 471
- NOT BETWEEN 495
- NOT EXISTS 495
- NOT IN 495
- NOT NULL 485
- Notationshinweise 149
- NSAPI 48
- NULL 169, 485
- number_format() 204
- O**
 - ob_end_clean() 667
 - ob_end_flush() 667
 - ob_get_contents() 667
 - ob_implicit_flush() 667
 - ob_start() 667
 - octdec() 205
 - ODBC 518
 - Architektur 520
 - Einführung 518
 - Funktionen 528
 - Treiber 519
 - und MySQL 527
 - unter Windows einrichten 521
 - odbc_connect() 580
 - odbc_exec() 581
 - odbc_fetch_into() 582
 - odbc_fetch_row() 582
 - odbc_pconnect() 580
 - odbc_result() 582
 - ODBC-Funktionen 572, 580
 - ODBC-Treiber 522
 - Oktett 63
 - Operatoren 170
 - : 262
 - ! 231
 - % 170
 - & 171
 - && 231
 - * 170
 - / 170
 - @ 273
 - | 171
 - || 231
 - ~ 171
 - + 170
 - ++ 170
 - << 172
 - => (array) 175
 - => (foreach) 245
 - >> 172
 - arithmetische 170
 - Bitweise 171
 - logische *Siehe* Logische Operatoren
 - Zeichenketten verbinden 171
 - Zuweisungen 171
- Optionale Parameter 248

- or **231**
- OR 495
- Oracle 519
- ord() **197**
- ORDER BY 491
- Oster-Funktionen *Siehe*
Kalenderfunktionen
- P**
- parse_str() 190, **201**
- parse_url() **228**
- Parser 817
- pclose() **355**
- PCRE *Siehe* Perl Compatible Regular
Expressions
- PDF *Siehe* Portable Document Format
- PDF-Funktionen 656
- pdflib 655
- Perl 47
- Perl Compatible Regular Expressions 669
 - Erweiterungen 670
 - Funktionen 671
- Persistente Verbindungen 544
- Personal Web Server *Siehe* Installation,
Windows (PWS)
- pfsockopen() **386**
- PHP
 - ausführen 147
 - Dateierweiterungen 147
 - einbinden, in HTML 148
 - Einführung 151
 - und HTML 147
 - und JavaScript 151
- PHP konfigurieren 119
- php.ini 119
- php.ini-dist 120
- PHP_OS 169
- PHP_VERSION 169
- php3.ini 119
- phpHoo* 553
- phpinfo()
 - Typische Ausgabe 139
- phpLIB 770
- phpMyAdmin 586
 - Installation 587
 - Konfiguration 587
 - Problembehandlung 590
 - Sprache einstellen 590
- PHP-Sessions *Siehe* Sessionfunktionen
- phpTemple 769
 - Bilderzeugung 795
 - Cache 800
 - Einschränkungen 805
 - Header 798
 - Installation 800
 - Konfiguration 801
 - Lizenzfragen 845
 - Parser 817
- Polymorphie 255
- popen() **355**
- Port 70
- Port- und Protokollnummern 69
- Portable Document Format 654
- Portbezeichnung 70
- pos() **184**
- POSIX 222
- POSIX-Ersatzsymbol 222
- POST 301
- pow() **205**
- preg_grep() **696**
- preg_match() **692**
- preg_match_all() **693**

preg_quote() 696
preg_replace() 695
preg_split() 696
prev() 183
PRIMARY KEY 485
print() 152
print_r() 285
printf() 201
Programmieren mit PHP 230
Proposed Standard 58
Protokoll
 verbindungsloses 73
Protokollnummern 71
Prozeduren 245
PUT 377

Q

Quoted Printable 96
quoted_printable_decode() 190
quotemeta() 197

R

rand() 207
range() 178
Rasterung 412
rawurldecode() 200
rawurlencode() 200
RDBMS 470
readfile() 356
readlink() 355
Realm 408
referenzierte Argumente (&) 247
REGEXP 496
register_shutdown_function() 335

Regular expressions *Siehe* Reguläre Ausdrücke
Reguläre Ausdrücke
 \$ 218
Reguläre Ausdrücke
 " 222
 () 220
 * 219
 . 221
 ? 219
 ^ 218
 {} 219
 | 220
 + 219
Alternativen 220
Backslash 222, 674
Bedingte Teilmuster 690
Bedingungen 687
Beispiele 223
Buchtipps 671
Effizienz 692
Einführung 217
Einmalige Teilmuster 689
Ersatzzeichen 223
Ersetzungen 694
Funktionen 226
Für Fortgeschrittene 669
Greedy 684
Gruppierung 220
Interne Optionen 680
Mehrzeilenmodus 677
Nongreedy 684
PCRE-Optionen 673
POSIX 222
Punkt 221
Referenzen 675
Rückwärts-Referenzen 685
Teilmuster 682
Unterschiede zu Perl 669

V
S
I
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E

- Whitespaces 674
- Wiederholung 219
- Wiederholungen 683
- Wortbeginn 218
- Wortende 218
- Zeichenklassen 676
- Zeichenklassendefinition 678
- Rekursion 251
- Relationen 471
- rename() **366**
- Request For Comments 57
 - Stufen 57
- Request-Header-Fields 75
- Request-URI 301
- require() **150**
- Reseau IP Europeens 66
- Reservierte Adressen 65
- reset() **184**
- Response-Header-Fields 75
- return() **246**
- REVOKE 515
- RFC *Siehe* Request For Comments
- RFC 1036 106
- RFC 1321 341
- RFC 1738 201
- RFC 1867 372
- RFC 1939 87
- RFC 1957 87
- RFC 2045 91, 229
- RFC 2046 91
- RFC 2068 229, 301
- RFC 2077 93
- RFC 2236 65
- RFC 2821 84
- RFC 2980 104

- RFC 822 91
- RFC 850 106
- RFC 977 104
- RFC1350 77
- RFC1639 83
- RFC1945 73
- RFC913 77
- RFC959 77
- RIPE NCC *Siehe* Reseau IP Europeens
- RLIKE 496
- rmdir() **355**
- ROT-13 99
- round() **205**
- rsort() **188**
- Rückgabe mehrerer Werte 251
- Rückgabewert einer Funktion 247

S

- Scharlatane* 401
- Schichtenmodell 55
- Schleifen 238
- Seiten schützen 406
- SELECT * FROM 487, 488
- SELECT ... WHERE 488
- SELECT INTO OUTFILE 511
- session_decode() **350**
- session_destroy() **348**
- session_encode() **350**
- session_get_cookie_params() **346**
- session_id() **345**
- session_is_registered() **345**
- session_module_name() **345**
- session_name() **348**
- session_register() **350**
- session_save_path() **349**

- session_set_cookie_params() 346
- session_start() 348
- session_unregister() 350
- Sessionfunktionen 345
- Sessions 341
 - konfigurieren 347
- Sessionvariablen 349
- Sessionverwaltung 336
 - Cookies 340
 - Methoden 337
 - URI 340
 - versteckte Felder 338
- Sessionverwaltung (phpTemple) 810
- SET 513
- set_socket_blocking() 385
- set_time_limit() 335
- setcookie() 230, 329
- setlocale() 215
- settype() 164
- SGML-Style 148
- SHOW 512
- shuffle() 178
- SHUTDOWN 514
- Sicherheit 135, 400
 - Angriffsszenarien 136
 - Kommandozeilenparameter 136
 - Parameterattacke 136
 - Pfadattacke 136
- Sicherheitsprobleme 135
- similar_text() 194
- Simple Mail Transfer Protocol 84
- sin() 205
- Sitemanagement 777
- Sitzungsverwaltung
 - Datenbanken 342
 - Textdateien 342
 - Zeitbegrenzung 343
- Skripte verbinden 318
- Skriptsprache 147
- Skriptsteuerung 271
- sleep() 336
- Smarty 770
- SMTP *Siehe* Simple Mail Transfer Protocol
 - Programmierung 642
- Socket 70, 71
- Socket-Funktionen 631
- Software für PHP 106
- Sonderzeichenbehandlung 197
- sort() 188
- Sortierfunktionen 178
 - für Arrays *Siehe* Arrayfunktionen
- soundex() 198
- Späte Bindung 246
- Spione 401
- split() 228
- sprintf() 201
- SQL *Siehe* Structured Query Language
 - Aggregate Funktionen 500
- sqrt() 205
- srand() 207
- sscanf() 203
- stat() 364
- static 158
- str_repeat() 193
- str_replace() 196
- STRAIGHT_JOIN 508
- strcasecmp() 198
- STRCMP 496
- strcmp() 198
- strcspn() 192, 196
- strftime() 213

strip_tags() 190
stripclashes() 189
stripslashes() 197, 321
strlen() 194
strpos() 195
strrchr() 195
strrev() 194
strrpos() 195
strspn() 192, 196
strstr() 195
strtok() 192
strtolower() 194
strtoupper() 194
strtr() 193, 296
Structured Query Language 468
Strukturen 230
Strukturierung 281
Subnetze 63
Subnetzmaske 63
substr() 195
Subtraktion 170
Suchmuster *Siehe* Reguläre Ausdrücke
SUM 501
switch() 236
 Erweiterte Syntax 237
symlink() 356
System-DSN 520

T

Tabellen
 ändern 486
 löschen 485
tan() 205
TCP
 Kapselung der Daten 56

TCP/IP 71
Telnet 50
Templatesysteme 769
tempname() 356
Textformatierung 278
this *Siehe* \$this
Thomas Boutell 411
Thread (NNTP) 106
time() 212
Timeout 334
touch() 356
Transmission Control Protocol
 Multiplexing 70
trim() 194
Trinärer Bedingungsoperator 235
TRUE 169
Typlosigkeit 157
Typumwandlung
 automatische 162
 explizite 163
 Typkonvertierung 163

U

ucfirst() 194
ucwords() 194
uksort() 188
umask() 356
Umgebungsvariablen 379
Umleitung 230
Umwandlungsfunktion 164
Umwandlungsfunktionen 197
Umwandlungsfunktionen, mathematische 205
uniqid() 342
Unisys 413

UNIX-Rechtesystem 401

unlink() 367

unset() 167

UPDATE 491

Upload *Siehe* Dateiupload

urldecode() 229, 320

urlencode() 229, 320

ursort() 188

User-DSN 520

USING 509

usleep() 336

usort() 188

UTF-8 703

utf8_decode() 704

utf8_encode() 704

UUencode 90

V

Vandalen 401

var_dump() 285

Variable Argumentelisten 249

Variablen 157

 dynamische 159

 Gültigkeitsbereiche 157

Verbindungen zu Servern 378

Verbindungssteuerung 333

Vererbung 255

Vergleiche 198

Vergleichsoperatoren 231, 232

Verschlüsselung

 der URL (phpTemple) 832

Verschlüsselungsfunktionen 197

Verzeichnisfunktionen 367

Verzeichnissystem 369

VI Improved (Editor) 750

Vorlage

 Verzweigungen 792

W

WAIS-Datenbank 52

WAMP *Siehe* Windows Apache MySQL
 PHP

WAP *Siehe* Wireless Application Protocol

WAP-Handy 726

 Emulatoren 726

WDDX *Siehe* Web Distributed Data
 Exchange

WDDX

 Komplexe Datentypen 739

WDDX SDK 736

wddx_add_vars() 742

wddx_deserialize() 742

wddx_packet_end() 742

wddx_packet_start() 741

wddx_serialize_value() 741

wddx_serialize_vars() 741

WDDX-Elemente 737

WDDX-Funktionen 741

Web Distributed Data Exchange 733

 Datentypen 738

 DTD 737

 XML 737

Web Syndication 734

Webauthor (Editor) 749

Webservervariablen *Siehe* Servervariablen

Website zum Buch 32

Webpace 137

Webspezifische Funktionen 228

Werte zuweisen 157

WHERE

 Operatoren 494

V

S

I

1

2

3

4

5

6

7

8

9

10

11

A

B

C

D

E

while **239**
 Alternative Syntax **241**
Whitespaces **194**
WIMP **110** *Siehe* Windows IIS MySQL
 PHP
Windows Apache MySQL PHP **107**
Windows IIS MySQL PHP **110**
Windows Xitami MySQL PHP **113**
Wireless Application Protocol **724**
Wireless Markup Language **724**
 Voraussetzungen **725**
Wireless Telephony Application Interface
 729
WML *Siehe* Wireless Markup Language
World Wide Web **50**
WTAI *Siehe* Wireless Telephony
 Application Interface
WWW *Siehe* World Wide Web
WXMP *Siehe* Windows Xitami MySQL
 PHP

X

Xemacs (Editor) **750**
Xitami *Siehe* Installation, Windows
 (Xitami)
XML *Siehe* Extensible Markup Language
 Case Folding **702**
 Fehlercodes **702**
 Funktionsübersicht **703**
 Zeichenkodierung **703**
xml_error_string() **703**
xml_get_current_byte_index() **704**
xml_get_current_column_number() **704**
xml_get_current_line_number() **704**

xml_get_error_code() **703**
xml_parse() **703**
xml_parser_create() **703**
xml_parser_free() **704**
xml_parser_get_option() **704**
xml_parser_set_option() **704**
XML-Style **148**
xor **231**
XSL *Siehe* XML

Z

Zählschleifen **241**
Zeichenketten
 Ersetzen von **196**
 suchen in **195**
Zeichenkettenformatierungen **201**
Zeichenkettenfunktionen **189**
 HTML-spezifische *Siehe* HTML-
 Funktionen
 Leerzeichen entfernen **194**
Zeichenketten-konvertierung **164**
Zeichenkettenmanipulation **194**
Zeichenkettenoperator **171**
Zeichenkettenverarbeitung **189**
Zeit und Datum **207**
Zeitfunktionen **212**
Zielgruppe **27**
Zufallszahlen **206**
Zufallszahlen-Funktionen **206**
Zugriffsrechte
 Webserver **404**
Zustände für Verbindungen **334**
Zuweisungsoperatoren **171**

E An den Autor

Für Fragen, Anregungen aber auch Kritik und Hinweise steht Ihnen der Autor gern zur Verfügung. Dieses Buch soll eine solide Basis für alle sein, die in der schnell wachsenden PHP-Gemeinde Fuß fassen wollen und ein solides, deutschsprachiges Arbeitsmittel benötigen. Insofern sind Verbesserungsvorschläge und konstruktive Anmerkungen gern gesehen und finden in künftigen Auflagen sicher ihre Entsprechung.

Aktuelle Informationen finden Sie im Internet

Alle Skripte, Bugfixes und Korrekturen finden Sie im Internet unter der folgenden Adresse:

<http://www.php.comzept.de/php4>

Mehr Informationen rund um PHP sind auf der *Partnersite zum Buch* zu entdecken:

<http://www.phparchiv.de>

Den Autor selbst können Sie auf seiner *Homepage* näher kennenlernen:

<http://www.joerg.krause.net>

Informationen über professionelle Unterstützung

Hilfe finden Sie beim Autor, wenn es um eines der folgenden »Probleme« geht:

- Entwicklung professioneller Websites jeder Größenordnung
- Projektmanagement und Programmierung in PHP, ASP, ASP.NET, JavaScript, HTML usw.
- Schulungen, Seminare, Programmierkurse und Workshops
- Fachliche Unterstützung für Start-Ups, Venture Capitalists und Old Economy ;-)

Anfragen senden Sie bitte direkt per E-Mail an:

joerg@krause.net

Leider schaffe ich es nicht immer, jede E-Mail sofort zu beantworten. Sie können aber sicher sein, dass jede E-Mail gelesen wird. Insofern können Sie dieses Medium jederzeit nutzen, um irgendetwas los zu werden – was auch immer Sie gern mitteilen möchten.